Good morning.

Randomized Algorithms, Lecture 10

Jacob Holm (jaho@di.ku.dk)

May 27th 2019

Today's Lecture

Algebraic Techniques

- Fingerprinting
- Freivald's Technique
  - Matrix Product Verification
  - Verifying Polynomial Identities
- Schwartz-Zippel Theorem
  - Application: Bipartite perfect matching
- String matching
  - I
  - II
  - I vs II
  - Pattern matching
- Summary

Fingerprinting

Suppose Alice has element $a \in U$ and Bob has element $b \in U$ for some large universe U, and Alice wants to send a message to Bob that lets him determine if $a = b$.

Deterministically, she needs to communciate at least $\log_2 |U|$ bits. Why?

Fingerprinting

Suppose Alice has element $a \in U$ and Bob has element $b \in U$ for some large universe U, and Alice wants to send a message to Bob that lets him determine if $a = b$.

Deterministically, she needs to communciate at least $\log_2 |U|$ bits. Why?

Fingerprinting

Suppose Alice has element $a \in U$ and Bob has element $b \in U$ for some large universe U, and Alice wants to send a message to Bob that lets him determine if $a = b$.

Deterministically, she needs to communciate at least $\log_2 |U|$ bits. Why?

Fingerprinting

Suppose Alice has element $a \in U$ and Bob has element $b \in U$ for some large universe U, and Alice wants to send a message to Bob that lets him determine if $a = b$.

Deterministically, she needs to communciate at least $\log_2 |U|$ bits. Why? Otherwise she will send the same message for some two different elements.

Fingerprinting

Alternatively, Alice can choose r random bits and use them to select a random *fingerprint function* $F : U \to S$. Alice then sends F and $F(a)$, and Bob can test if $F(a) = F(b)$. This takes $\mathcal{O}(r + \log_2 |S|)$ bits, but risks *false positives*.

Very specialized version of hashing.

We want $r, |S|$ small, we want F fast, and we want $\Pr_F[F(a) = F(b) \mid a \neq b]$ to be small.

Let $\mathbb{F}$ be a field (e.g. $\mathbb{Q}$, or $\mathbb{Z}_p$ for some prime p).

Assume each operation on $\mathbb{F}$ takes one unit of time.

Fingerprinting

Alternatively, Alice can choose r random bits and use them to select a random *fingerprint function* $F : U \to S$. Alice then sends F and $F(a)$, and Bob can test if $F(a) = F(b)$. This takes $\mathcal{O}(r + \log_2|S|)$ bits, but risks *false positives*.

Very specialized version of hashing.

We want $r, |S|$ small, we want F fast, and we want $\Pr_F[F(a) = F(b) \mid a \neq b]$ to be small.

Let $\mathbb{F}$ be a field (e.g. $\mathbb{Q}$, or $\mathbb{Z}_p$ for some prime p).

Assume each operation on $\mathbb{F}$ takes one unit of time.

Fingerprinting

Alternatively, Alice can choose r random bits and use them to select a random *fingerprint function* $F : U \to S$. Alice then sends F and $F(a)$, and Bob can test if $F(a) = F(b)$. This takes $\mathcal{O}(r + \log_2|S|)$ bits, but risks *false positives.*

Very specialized version of hashing.

We want $r, |S|$ small, we want F fast, and we want $\Pr_F[F(a) = F(b) \mid a \neq b]$ to be small.

Let $\mathbb{F}$ be a field (e.g. $\mathbb{Q}$, or $\mathbb{Z}_p$ for some prime p).

Assume each operation on $\mathbb{F}$ takes one unit of time.

Fingerprinting

Alternatively, Alice can choose r random bits and use them to select a random *fingerprint function* $F : U \to S$. Alice then sends F and $F(a)$, and Bob can test if $F(a) = F(b)$. This takes $\mathcal{O}(r + \log_2 |S|)$ bits, but risks *false positives*.

Very specialized version of hashing.

We want $r, |S|$ small, we want F fast, and we want $\Pr_F[F(a) = F(b) \mid a \neq b]$ to be small.

Let $\mathbb{F}$ be a field (e.g. $\mathbb{Q}$, or $\mathbb{Z}_p$ for some prime p).

Assume each operation on $\mathbb{F}$ takes one unit of time.

Fingerprinting

Alternatively, Alice can choose r random bits and use them to select a random *fingerprint function* $F : U \to S$. Alice then sends F and $F(a)$, and Bob can test if $F(a) = F(b)$. This takes $\mathcal{O}(r + \log_2 |S|)$ bits, but risks *false positives*.

Very specialized version of hashing.

We want $r, |S|$ small, we want F fast, and we want $\Pr_F[F(a) = F(b) \mid a \neq b]$ to be small.

Let $\mathbb{F}$ be a field (e.g. $\mathbb{Q}$, or $\mathbb{Z}_p$ for some prime p).

Assume each operation on $\mathbb{F}$ takes one unit of time.

Fingerprinting

Alternatively, Alice can choose r random bits and use them to select a random *fingerprint function* $F : U \to S$. Alice then sends F and $F(a)$, and Bob can test if $F(a) = F(b)$. This takes $\mathcal{O}(r + \log_2|S|)$ bits, but risks *false positives*.

Very specialized version of hashing.

We want $r, |S|$ small, we want F fast, and we want $\Pr_F[F(a) = F(b) \mid a \neq b]$ to be small.

Let $\mathbb{F}$ be a field (e.g. $\mathbb{Q}$, or $\mathbb{Z}_p$ for some prime p).

Assume each operation on $\mathbb{F}$ takes one unit of time.

Fingerprinting

Alternatively, Alice can choose r random bits and use them to select a random *fingerprint function* $F : U \to S$. Alice then sends F and $F(a)$, and Bob can test if $F(a) = F(b)$. This takes $\mathcal{O}(r + \log_2|S|)$ bits, but risks *false positives*.

Very specialized version of hashing.

We want $r, |S|$ small, we want F fast, and we want $\Pr_F[F(a) = F(b) \mid a \neq b]$ to be small.

Let $\mathbb{F}$ be a field (e.g. $\mathbb{Q}$, or $\mathbb{Z}_p$ for some prime p).

Assume each operation on $\mathbb{F}$ takes one unit of time.

A field is a set $\mathbb{F}$ with operations $+$ and $\cdot$ and special values $0, 1 \in \mathbb{F}$, that essentially behave as they do for the rational numbers. This includes the existence of a unique *additive inverse* $-a$ for each element $a \in \mathbb{F}$, and a unique multiplicative inverse a^{-1} for each $a \in \mathbb{F} \setminus \{0\}$.
For this talk, it is enough to know that $\mathbb{Z}_p$ is a field for any prime p, and that e.g. a polynomial of degree d over $\mathbb{Z}_p$ has at most d roots.

Fingerprinting

Alternatively, Alice can choose r random bits and use them to select a random *fingerprint function* $F : U \to S$. Alice then sends F and $F(a)$, and Bob can test if $F(a) = F(b)$. This takes $\mathcal{O}(r + \log_2|S|)$ bits, but risks *false positives*.

Very specialized version of hashing.

We want $r, |S|$ small, we want F fast, and we want $\Pr_F[F(a) = F(b) \mid a \neq b]$ to be small.

Let $\mathbb{F}$ be a field (e.g. $\mathbb{Q}$, or $\mathbb{Z}_p$ for some prime p).

Assume each operation on $\mathbb{F}$ takes one unit of time.

In other words, when we talk about time we'll really be counting number of field operations. For the problems and fields we are interested in, this makes very little difference.

Freivald's Technique: Matrix product verif.

Given matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{F}^{n \times n}$. Suppose some algorithm (Alice) claims to have computed $\mathbf{AB}$ and the result is $\mathbf{C}$. How do we (Bob) verify this?

Naively, we could just compute $\mathbf{AB}$ using e.g. fast matrix multiplication in $\mathcal{O}(n^{2.3728639})$ time, and compare. Why is this not a good idea?

Freivald's Technique: Matrix product verif.

Given matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{F}^{n \times n}$. Suppose some algorithm (Alice) claims to have computed $\mathbf{AB}$ and the result is $\mathbf{C}$. How do we (Bob) verify this?

Naively, we could just compute $\mathbf{AB}$ using e.g. fast matrix multiplication in $\mathcal{O}(n^{2.3728639})$ time, and compare. Why is this not a good idea?

Freivald's Technique: Matrix product verif.

Given matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{F}^{n \times n}$. Suppose some algorithm (Alice) claims to have computed $\mathbf{AB}$ and the result is $\mathbf{C}$. How do we (Bob) verify this?

Naively, we could just compute $\mathbf{AB}$ using e.g. fast matrix multiplication in $\mathcal{O}(n^{2.3728639})$ time, and compare. Why is this not a good idea?

Freivald's Technique: Matrix product verif.

Given matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{F}^{n \times n}$. Suppose some algorithm (Alice) claims to have computed $\mathbf{AB}$ and the result is $\mathbf{C}$. How do we (Bob) verify this?

Naively, we could just compute $\mathbf{AB}$ using e.g. fast matrix multiplication in $\mathcal{O}(n^{2.3728639})$ time, and compare. Why is this not a good idea?

Freivald's Technique: Matrix product verif.

Given matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{F}^{n \times n}$. Suppose some algorithm (Alice) claims to have computed $\mathbf{AB}$ and the result is $\mathbf{C}$. How do we (Bob) verify this?

Naively, we could just compute $\mathbf{AB}$ using e.g. fast matrix multiplication in $\mathcal{O}(n^{2.3728639})$ time, and compare. Why is this not a good idea?

Freivald's Technique: Matrix product verif.

Given matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{F}^{n \times n}$. Suppose some algorithm (Alice) claims to have computed $\mathbf{AB}$ and the result is $\mathbf{C}$. How do we (Bob) verify this?

Naively, we could just compute $\mathbf{AB}$ using e.g. fast matrix multiplication in $\mathcal{O}(n^{2.3728639})$ time, and compare. Why is this not a good idea? Very complicated. What if we made mistake?

Freivald's Technique: Matrix product verif.

Theorem

For $\mathbf{r} \in \{0,1\}^n$ *chosen uniformly at random,*

$$\Pr_{\mathbf{r}}[\mathbf{A}(\mathbf{B}\mathbf{r}) = \mathbf{C}\mathbf{r} \mid \mathbf{A}\mathbf{B} \neq \mathbf{C}] \leq \frac{1}{2}$$

We can compute $\mathbf{x} = \mathbf{B}\mathbf{r}$, $\mathbf{y} = \mathbf{A}\mathbf{x}$, and $\mathbf{z} = \mathbf{C}\mathbf{r}$ in $\mathcal{O}(n^2)$ time each and compare $\mathbf{y} = \mathbf{z}$ in $\mathcal{O}(n)$ time. So in $\mathcal{O}(tn^2)$ time we can get error probability 2^{-t}.

The idea here is that $F(\mathbf{AB}) = (\mathbf{AB})\mathbf{r} = \mathbf{A}(\mathbf{B}\mathbf{r})$ can be computed faster than computing $\mathbf{AB}$.

Freivald's Technique: Matrix product verif.

Theorem

For $\mathbf{r} \in \{0,1\}^n$ *chosen uniformly at random,*

$$\Pr_{\mathbf{r}}[\mathbf{A}(\mathbf{B}\mathbf{r}) = \mathbf{C}\mathbf{r} \mid \mathbf{AB} \neq \mathbf{C}] \leq \frac{1}{2}$$

We can compute $\mathbf{x} = \mathbf{B}\mathbf{r}$, $\mathbf{y} = \mathbf{A}\mathbf{x}$, and $\mathbf{z} = \mathbf{C}\mathbf{r}$ in $\mathcal{O}(n^2)$ time each and compare $\mathbf{y} = \mathbf{z}$ in $\mathcal{O}(n)$ time. So in $\mathcal{O}(tn^2)$ time we can get error probability 2^{-t}.

The idea here is that $F(\mathbf{AB}) = (\mathbf{AB})\mathbf{r} = \mathbf{A}(\mathbf{B}\mathbf{r})$ can be computed faster than computing $\mathbf{AB}$.

Freivald's Technique: Matrix product verif.

Theorem

For $\mathbf{r} \in \{0,1\}^n$ *chosen uniformly at random,*

$$\Pr_{\mathbf{r}}[\mathbf{A}(\mathbf{Br}) = \mathbf{Cr} \mid \mathbf{AB} \neq \mathbf{C}] \leq \frac{1}{2}$$

We can compute $\mathbf{x} = \mathbf{Br}$, $\mathbf{y} = \mathbf{Ax}$, and $\mathbf{z} = \mathbf{Cr}$ in $\mathcal{O}(n^2)$ time each and compare $\mathbf{y} = \mathbf{z}$ in $\mathcal{O}(n)$ time.
So in $\mathcal{O}(tn^2)$ time we can get error probability 2^{-t}.

The idea here is that $F(\mathbf{AB}) = (\mathbf{AB})\mathbf{r} = \mathbf{A}(\mathbf{Br})$ can be computed faster than computing $\mathbf{AB}$.

Freivald's Technique: Matrix product verif.

Proof.

Suppose $\mathbf{AB} \neq \mathbf{C}$, and let $\mathbf{D} = \mathbf{AB} - \mathbf{C}$. Then $\mathbf{D} \neq 0$, so we can choose i, j so $\mathbf{D}_{ij} \neq 0$. Now

$$\begin{aligned}
\Pr_{\mathbf{r}}[\mathbf{A}(\mathbf{Br}) = \mathbf{Cr}] &= \Pr_{\mathbf{r}}[\mathbf{Dr} = 0] \\
&\leq \Pr_{\mathbf{r}}[\mathbf{D}_i \mathbf{r} = 0] \\
&= \Pr_{\mathbf{r}}\left[\sum_k \mathbf{D}_{ik}\mathbf{r}_k = 0\right] \\
&= \Pr_{\mathbf{r}}\left[\mathbf{r}_j = \frac{-1}{\mathbf{D}_{ij}} \sum_{k \neq j} \mathbf{D}_{ik}\mathbf{r}_k\right] \\
&\leq \frac{1}{2}
\end{aligned}$$

$\square$

Freivald's Technique: Matrix product verif.

Proof.

Suppose $\mathbf{AB} \neq \mathbf{C}$, and let $\mathbf{D} = \mathbf{AB} - \mathbf{C}$. Then $\mathbf{D} \neq 0$, so we can choose i, j so $\mathbf{D}_{ij} \neq 0$. Now

$$\begin{aligned}
\Pr_{\mathbf{r}}[\mathbf{A}(\mathbf{Br}) = \mathbf{Cr}] &= \Pr_{\mathbf{r}}[\mathbf{Dr} = 0] \\
&\leq \Pr_{\mathbf{r}}[\mathbf{D}_i\mathbf{r} = 0] \\
&= \Pr_{\mathbf{r}}\left[\sum_k \mathbf{D}_{ik}\mathbf{r}_k = 0\right] \\
&= \Pr_{\mathbf{r}}\left[\mathbf{r}_j = \tfrac{-1}{\mathbf{D}_{ij}} \sum_{k \neq j} \mathbf{D}_{ik}\mathbf{r}_k\right] \\
&\leq \tfrac{1}{2}
\end{aligned}$$

□

Freivald's Technique: Matrix product verif.

$$\mathbf{A}(\mathbf{B}\mathbf{r}) = \mathbf{C}\mathbf{r} \iff (\mathbf{A}\mathbf{B})\mathbf{r} - \mathbf{C}\mathbf{r} = 0 \iff (\mathbf{A}\mathbf{B} - \mathbf{C})\mathbf{r} = 0$$

Proof.

Suppose $\mathbf{AB} \neq \mathbf{C}$, and let $\mathbf{D} = \mathbf{AB} - \mathbf{C}$. Then $\mathbf{D} \neq 0$, so we can choose i, j so $\mathbf{D}_{ij} \neq 0$. Now

$$\begin{aligned}
\Pr_{\mathbf{r}}[\mathbf{A}(\mathbf{B}\mathbf{r}) = \mathbf{C}\mathbf{r}] &= \Pr_{\mathbf{r}}[\mathbf{D}\mathbf{r} = 0] \\
&\leq \Pr_{\mathbf{r}}[\mathbf{D}_i\mathbf{r} = 0] \\
&= \Pr_{\mathbf{r}}\left[\sum_k \mathbf{D}_{ik}\mathbf{r}_k = 0\right] \\
&= \Pr_{\mathbf{r}}\left[\mathbf{r}_j = \frac{-1}{\mathbf{D}_{ij}} \sum_{k \neq j} \mathbf{D}_{ik}\mathbf{r}_k\right] \\
&\leq \frac{1}{2}
\end{aligned}$$

□

Since $\mathbf{Dr} = 0 \iff \forall \ell : \mathbf{D}_\ell \mathbf{r} = 0 \implies \mathbf{D}_i \mathbf{r} = 0$.

Freivald's Technique: Matrix product verif.

Proof.

Suppose $\mathbf{AB} \neq \mathbf{C}$, and let $\mathbf{D} = \mathbf{AB} - \mathbf{C}$. Then $\mathbf{D} \neq 0$, so we can choose i, j so $\mathbf{D}_{ij} \neq 0$. Now

$$\begin{aligned}
\Pr_{\mathbf{r}}[\mathbf{A}(\mathbf{Br}) = \mathbf{Cr}] &= \Pr_{\mathbf{r}}[\mathbf{Dr} = 0] \\
&\leq \Pr_{\mathbf{r}}[\mathbf{D}_i \mathbf{r} = 0] \\
&= \Pr_{\mathbf{r}}\left[\sum_k \mathbf{D}_{ik} \mathbf{r}_k = 0\right] \\
&= \Pr_{\mathbf{r}}\left[\mathbf{r}_j = \frac{-1}{\mathbf{D}_{ij}} \sum_{k \neq j} \mathbf{D}_{ik} \mathbf{r}_k\right] \\
&\leq \frac{1}{2}
\end{aligned}$$

$\square$

Freivald's Technique: Matrix product verif.

Proof.

Suppose $\mathbf{AB} \neq \mathbf{C}$, and let $\mathbf{D} = \mathbf{AB} - \mathbf{C}$. Then $\mathbf{D} \neq 0$, so we can choose i, j so $\mathbf{D}_{ij} \neq 0$. Now

$$\begin{aligned}
\Pr_{\mathbf{r}}[\mathbf{A}(\mathbf{Br}) = \mathbf{Cr}] &= \Pr_{\mathbf{r}}[\mathbf{Dr} = 0] \\
&\leq \Pr_{\mathbf{r}}[\mathbf{D}_i\mathbf{r} = 0] \\
&= \Pr_{\mathbf{r}}\left[\sum_k \mathbf{D}_{ik}\mathbf{r}_k = 0\right] \\
&= \Pr_{\mathbf{r}}\left[\mathbf{r}_j = \frac{-1}{\mathbf{D}_{ij}} \sum_{k \neq j} \mathbf{D}_{ik}\mathbf{r}_k\right] \\
&\leq \frac{1}{2} \qquad \square
\end{aligned}$$

Freivald's Technique: Matrix product verif.

Proof.

Suppose $\mathbf{AB} \neq \mathbf{C}$, and let $\mathbf{D} = \mathbf{AB} - \mathbf{C}$. Then $\mathbf{D} \neq 0$, so we can choose i, j so $\mathbf{D}_{ij} \neq 0$. Now

$$\begin{aligned}
\Pr_{\mathbf{r}}[\mathbf{A}(\mathbf{Br}) = \mathbf{Cr}] &= \Pr_{\mathbf{r}}[\mathbf{Dr} = 0] \\
&\leq \Pr_{\mathbf{r}}[\mathbf{D}_i\mathbf{r} = 0] \\
&= \Pr_{\mathbf{r}}\left[\sum_k \mathbf{D}_{ik}\mathbf{r}_k = 0\right] \\
&= \Pr_{\mathbf{r}}\left[\mathbf{r}_j = \tfrac{-1}{\mathbf{D}_{ij}} \sum_{k \neq j} \mathbf{D}_{ik}\mathbf{r}_k\right] \\
&\leq \tfrac{1}{2}
\end{aligned}$$

$\square$

This is valid since $\mathbf{D}_{ij} \neq 0$. Simply subtract all terms except $\mathbf{D}_{ij}\mathbf{r}_j$ on both sides, and divide by $\mathbf{D}_{ij}$.

Freivald's Technique: Matrix product verif.

Proof.

Suppose $\mathbf{AB} \neq \mathbf{C}$, and let $\mathbf{D} = \mathbf{AB} - \mathbf{C}$. Then $\mathbf{D} \neq 0$, so we can choose i, j so $\mathbf{D}_{ij} \neq 0$. Now

$$\begin{aligned}
\Pr_{\mathbf{r}}[\mathbf{A}(\mathbf{Br}) = \mathbf{Cr}] &= \Pr_{\mathbf{r}}[\mathbf{Dr} = 0] \\
&\leq \Pr_{\mathbf{r}}[\mathbf{D}_i\mathbf{r} = 0] \\
&= \Pr_{\mathbf{r}}\left[\sum_k \mathbf{D}_{ik}\mathbf{r}_k = 0\right] \\
&= \Pr_{\mathbf{r}}\left[\mathbf{r}_j = \tfrac{-1}{\mathbf{D}_{ij}} \sum_{k \neq j} \mathbf{D}_{ik}\mathbf{r}_k\right] \\
&\leq \tfrac{1}{2}
\end{aligned}$$

$\square$

Uses the *principle of deferred decisions*. All entries in $\mathbf{r}$ are independent, so we can pretend that $\mathbf{r}_j$ is chosen last. Since it is chosen uniformly from $\{0, 1\}$ (a set with 2 distinct values), the chance of hitting the specific value $\frac{-1}{\mathbf{D}_{ij}} \sum_{k \neq j} \mathbf{D}_{ik}\mathbf{r}_k$ (whatever that is) is at most $\frac{1}{2}$.

Freivald's Technique: Polynomial ident.

Given polynomials $P_1(x), P_2(x) \in \mathbb{F}[x]$ of degree $\leq d$ as black boxes. How do we check if $P_1 = P_2$?

Theorem

Let $Q \in \mathbb{F}[x]$ have degree d, let $\mathbb{S} \subseteq \mathbb{F}$ be finite and $|\mathbb{S}| \geq d+1$. For $x \in \mathbb{S}$ picked uniformly at random,

$$\Pr_x[Q(x) = 0 \mid Q \neq 0] \leq \frac{d}{|\mathbb{S}|}$$

Choosing $Q = P_1 - P_2$, and $\mathbb{S}$ so $|\mathbb{S}| \geq 2d$ and repeating t times, then gives error probability $\leq 2^{-t}$. The idea is that we can compute $F(P_1 - P_2) = (P_1 - P_2)(x) = P_1(x) - P_2(x)$ without computing $P_1 - P_2$.

Freivald's Technique: Polynomial ident.

Given polynomials $P_1(x), P_2(x) \in \mathbb{F}[x]$ of degree $\leq d$ as black boxes. How do we check if $P_1 = P_2$?

Theorem

Let $Q \in \mathbb{F}[x]$ have degree d, let $\mathbb{S} \subseteq \mathbb{F}$ be finite and $|\mathbb{S}| \geq d + 1$. For $x \in \mathbb{S}$ picked uniformly at random,

$$\Pr_x[Q(x) = 0 \mid Q \neq 0] \leq \frac{d}{|\mathbb{S}|}$$

Choosing $Q = P_1 - P_2$, and $\mathbb{S}$ so $|\mathbb{S}| \geq 2d$ and repeating t times, then gives error probability $\leq 2^{-t}$.
The idea is that we can compute $F(P_1 - P_2) = (P_1 - P_2)(x) = P_1(x) - P_2(x)$ without computing $P_1 - P_2$.

Freivald's Technique: Polynomial ident.

Given polynomials $P_1(x), P_2(x) \in \mathbb{F}[x]$ of degree $\leq d$ as black boxes. How do we check if $P_1 = P_2$?

Theorem

Let $Q \in \mathbb{F}[x]$ have degree d, let $\mathbb{S} \subseteq \mathbb{F}$ be finite and $|\mathbb{S}| \geq d+1$. For $x \in \mathbb{S}$ picked uniformly at random,

$$\Pr_x[Q(x) = 0 \mid Q \neq 0] \leq \tfrac{d}{|\mathbb{S}|}$$

Choosing $Q = P_1 - P_2$, and $\mathbb{S}$ so $|\mathbb{S}| \geq 2d$ and repeating t times, then gives error probability $\leq 2^{-t}$. The idea is that we can compute $F(P_1 - P_2) = (P_1 - P_2)(x) = P_1(x) - P_2(x)$ without computing $P_1 - P_2$.

Freivald's Technique: Polynomial ident.

Given polynomials $P_1(x), P_2(x) \in \mathbb{F}[x]$ of degree $\leq d$ as black boxes. How do we check if $P_1 = P_2$?

Theorem

Let $Q \in \mathbb{F}[x]$ have degree d, let $\mathbb{S} \subseteq \mathbb{F}$ be finite and $|\mathbb{S}| \geq d + 1$. For $x \in \mathbb{S}$ picked uniformly at random,

$$\Pr_x[Q(x) = 0 \mid Q \neq 0] \leq \tfrac{d}{|\mathbb{S}|}$$

Choosing $Q = P_1 - P_2$, and $\mathbb{S}$ so $|\mathbb{S}| \geq 2d$ and repeating t times, then gives error probability $\leq 2^{-t}$.

The idea is that we can compute $F(P_1 - P_2) = (P_1 - P_2)(x) = P_1(x) - P_2(x)$ without computing $P_1 - P_2$.

Freivald's Technique: Polynomial ident.

Given polynomials $P_1(x), P_2(x) \in \mathbb{F}[x]$ of degree $\leq d$ as black boxes. How do we check if $P_1 = P_2$?

Theorem

Let $Q \in \mathbb{F}[x]$ have degree d, let $\mathbb{S} \subseteq \mathbb{F}$ be finite and $|\mathbb{S}| \geq d+1$. For $x \in \mathbb{S}$ picked uniformly at random,

$$\Pr_x[Q(x) = 0 \mid Q \neq 0] \leq \frac{d}{|\mathbb{S}|}$$

Choosing $Q = P_1 - P_2$, and $\mathbb{S}$ so $|\mathbb{S}| \geq 2d$ and repeating t times, then gives error probability $\leq 2^{-t}$.
The idea is that we can compute
$F(P_1 - P_2) = (P_1 - P_2)(x) = P_1(x) - P_2(x)$
without computing $P_1 - P_2$.

Freivald's Technique: Polynomial ident.

Proof.

A nonzero polynomial of degree d over any field has at most d distinct roots. Thus, $\mathbb{S}$ contains at most d roots, and the probability of picking one of them is at most $\frac{d}{|\mathbb{S}|}$. $\square$

Freivald's Technique: Polynomial ident.

Proof.

A nonzero polynomial of degree d over any field has at most d distinct roots. Thus, $\mathbb{S}$ contains at most d roots, and the probability of picking one of them is at most $\frac{d}{|\mathbb{S}|}$. $\square$

Freivald's Technique: Polynomial ident.

Proof.

A nonzero polynomial of degree d over any field has at most d distinct roots. Thus, $\mathbb{S}$ contains at most d roots, and the probability of picking one of them is at most $\frac{d}{|\mathbb{S}|}$. $\square$

Schwartz-Zippel Theorem

We can generalize this theorem to the multivariate case. Define the degree of $x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n}$ to be $d_1 + d_2 + \cdots + d_n$, and the *total degree* of a polynomial to be the maximum degree of any of its terms.

Theorem (Schwartz-Zippel)

Let $Q(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ have total degree d. Fix any finite set $\mathbb{S} \subseteq \mathbb{F}$ and let $r_1, \ldots, r_n$ be chosen independently and uniformly at random from $\mathbb{S}$. Then

$$\Pr[Q(r_1, \ldots, r_n) = 0 \mid Q \neq 0] \leq \frac{d}{|\mathbb{S}|}$$

Schwartz-Zippel Theorem

We can generalize this theorem to the multivariate case. Define the degree of $x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n}$ to be $d_1 + d_2 + \cdots + d_n$, and the *total degree* of a polynomial to be the maximum degree of any of its terms.

Theorem (Schwartz-Zippel)

Let $Q(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ have total degree d. Fix any finite set $\mathbb{S} \subseteq \mathbb{F}$ and let $r_1, \ldots, r_n$ be chosen independently and uniformly at random from $\mathbb{S}$. Then

$$\Pr[Q(r_1, \ldots, r_n) = 0 \mid Q \neq 0] \leq \frac{d}{|\mathbb{S}|}$$

Schwartz-Zippel Theorem

We can generalize this theorem to the multivariate case. Define the degree of $x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n}$ to be $d_1 + d_2 + \cdots + d_n$, and the *total degree* of a polynomial to be the maximum degree of any of its terms.

Theorem (Schwartz-Zippel)

Let $Q(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ have total degree d. Fix any finite set $\mathbb{S} \subseteq \mathbb{F}$ and let $r_1, \ldots, r_n$ be chosen independently and uniformly at random from $\mathbb{S}$. Then

$$\Pr[Q(r_1, \ldots, r_n) = 0 \mid Q \neq 0] \leq \frac{d}{|\mathbb{S}|}$$

Schwartz-Zippel Theorem

We can generalize this theorem to the multivariate case. Define the degree of $x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n}$ to be $d_1 + d_2 + \cdots + d_n$, and the *total degree* of a polynomial to be the maximum degree of any of its terms.

Theorem (Schwartz-Zippel)

Let $Q(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ have total degree d. Fix any finite set $\mathbb{S} \subseteq \mathbb{F}$ and let $r_1, \ldots, r_n$ be chosen independently and uniformly at random from $\mathbb{S}$. Then

$$\Pr[Q(r_1, \ldots, r_n) = 0 \mid Q \neq 0] \leq \frac{d}{|\mathbb{S}|}$$

Schwartz-Zippel Theorem

Proof by induction on n.

We have already proven $n = 1$. Assume $n \geq 2$ and that it holds for all smaller n. Let $Q \neq 0$, and let $k > 0$ be max exponent of x_n, then there exists $Q_0, \ldots, Q_k$ such that

$$Q(x_1, \ldots, x_n) = \sum_{i=0}^{k} Q_i(x_1, \ldots, x_{n-1}) x_n^i$$

By construction, $Q_k \neq 0$ and $\deg(Q_k) \leq d - k$. Pick $r_1, \ldots, r_{n-1}$ independently and uniformly at random from $\mathbb{S}$. Let $C_i = Q_i(r_1, \ldots, r_{n-1})$. By induction, $\Pr[C_k = 0] \leq \frac{d-k}{|\mathbb{S}|}$. If $C_k \neq 0$, then $q(x) = \sum_{i=1}^{k} C_i x^i = Q(r_1, \ldots, r_{n-1}, x) \neq 0$ has degree k, so for uniformly random $r_n \in \mathbb{S}$, $\Pr[q(r_n) = 0 \mid C_k \neq 0] \leq \frac{k}{|\mathbb{S}|}$. Finally,

$$\begin{aligned} \Pr[Q(r_1, \ldots, r_n) = 0] &\leq \Pr[C_k = 0] + \Pr[q(r_n) = 0 \mid C_k \neq 0] \\ &\leq \frac{d-k}{|\mathbb{S}|} + \frac{k}{|\mathbb{S}|} = \frac{d}{|\mathbb{S}|} \end{aligned}$$

$\square$

This is the previous theorem for univariate polynomials.

Schwartz-Zippel Theorem

Proof by induction on n.

We have already proven $n = 1$. Assume $n \geq 2$ and that it holds for all smaller n. Let $Q \neq 0$, and let $k > 0$ be max exponent of x_n, then there exists $Q_0, \ldots, Q_k$ such that

$$Q(x_1, \ldots, x_n) = \sum_{i=0}^{k} Q_i(x_1, \ldots, x_{n-1}) x_n^i$$

By construction, $Q_k \neq 0$ and $\deg(Q_k) \leq d - k$. Pick $r_1, \ldots, r_{n-1}$ independently and uniformly at random from $\mathbb{S}$. Let $C_i = Q_i(r_1, \ldots, r_{n-1})$. By induction, $\Pr[C_k = 0] \leq \frac{d-k}{|\mathbb{S}|}$. If $C_k \neq 0$, then $q(x) = \sum_{i=1}^{k} C_i x^i = Q(r_1, \ldots, r_{n-1}, x) \neq 0$ has degree k, so for uniformly random $r_n \in \mathbb{S}$, $\Pr[q(r_n) = 0 \mid C_k \neq 0] \leq \frac{k}{|\mathbb{S}|}$. Finally,

$$\begin{aligned} \Pr[Q(r_1, \ldots, r_n) = 0] &\leq \Pr[C_k = 0] + \Pr[q(r_n) = 0 \mid C_k \neq 0] \\ &\leq \frac{d-k}{|\mathbb{S}|} + \frac{k}{|\mathbb{S}|} = \frac{d}{|\mathbb{S}|} \end{aligned}$$

$\square$

Schwartz-Zippel Theorem

Proof by induction on n.

We have already proven $n = 1$. Assume $n \geq 2$ and that it holds for all smaller n. Let $Q \neq 0$, and let $k > 0$ be max exponent of x_n, then there exists $Q_0, \ldots, Q_k$ such that

$$Q(x_1, \ldots, x_n) = \sum_{i=0}^{k} Q_i(x_1, \ldots, x_{n-1}) x_n^i$$

By construction, $Q_k \neq 0$ and $\deg(Q_k) \leq d - k$. Pick $r_1, \ldots, r_{n-1}$ independently and uniformly at random from $\mathbb{S}$. Let $C_i = Q_i(r_1, \ldots, r_{n-1})$. By induction, $\Pr[C_k = 0] \leq \frac{d-k}{|\mathbb{S}|}$. If $C_k \neq 0$, then $q(x) = \sum_{i=1}^{k} C_i x^i = Q(r_1, \ldots, r_{n-1}, x) \neq 0$ has degree k, so for uniformly random $r_n \in \mathbb{S}$, $\Pr[q(r_n) = 0 \mid C_k \neq 0] \leq \frac{k}{|\mathbb{S}|}$. Finally,

$$\begin{aligned}\Pr[Q(r_1, \ldots, r_n) = 0] &\leq \Pr[C_k = 0] + \Pr[q(r_n) = 0 \mid C_k \neq 0] \\ &\leq \frac{d-k}{|\mathbb{S}|} + \frac{k}{|\mathbb{S}|} = \frac{d}{|\mathbb{S}|}\end{aligned}$$

□

If $Q = 0$ there is nothing to prove.

Schwartz-Zippel Theorem

Proof by induction on n.

We have already proven $n = 1$. Assume $n \geq 2$ and that it holds for all smaller n. Let $Q \neq 0$, and let $k > 0$ be max exponent of x_n, then there exists $Q_0, \ldots, Q_k$ such that

$$Q(x_1, \ldots, x_n) = \sum_{i=0}^{k} Q_i(x_1, \ldots, x_{n-1}) x_n^i$$

By construction, $Q_k \neq 0$ and $\deg(Q_k) \leq d - k$. Pick $r_1, \ldots, r_{n-1}$ independently and uniformly at random from $\mathbb{S}$. Let $C_i = Q_i(r_1, \ldots, r_{n-1})$. By induction, $\Pr[C_k = 0] \leq \frac{d-k}{|\mathbb{S}|}$.
If $C_k \neq 0$, then $q(x) = \sum_{i=1}^{k} C_i x^i = Q(r_1, \ldots, r_{n-1}, x) \neq 0$ has degree k, so for uniformly random $r_n \in \mathbb{S}$, $\Pr[q(r_n) = 0 \mid C_k \neq 0] \leq \frac{k}{|\mathbb{S}|}$. Finally,

$$\begin{aligned}\Pr[Q(r_1, \ldots, r_n) = 0] &\leq \Pr[C_k = 0] + \Pr[q(r_n) = 0 \mid C_k \neq 0] \\ &\leq \frac{d-k}{|\mathbb{S}|} + \frac{k}{|\mathbb{S}|} = \frac{d}{|\mathbb{S}|}\end{aligned}$$ □

If $k = 0$, that means x_n has no effect on the value of the polynomial. In particular, there exists an $(n-1)$-variate polynomial $R \neq 0$, such that $Q(x_1, \ldots, x_{n-1}, x_n) = R(x_1, \ldots, x_{n-1})$, so by induction $\Pr[Q(r_1, \ldots, r_n) = 0] = \Pr[R(r_1, \ldots, r_{n-1}) = 0] \leq \frac{d}{|\mathbb{S}|}$.

Schwartz-Zippel Theorem

Proof by induction on n.

We have already proven $n = 1$. Assume $n \geq 2$ and that it holds for all smaller n. Let $Q \neq 0$, and let $k > 0$ be max exponent of x_n, then there exists $Q_0, \ldots, Q_k$ such that

$$Q(x_1, \ldots, x_n) = \sum_{i=0}^{k} Q_i(x_1, \ldots, x_{n-1}) x_n^i$$

By construction, $Q_k \neq 0$ and $\deg(Q_k) \leq d - k$. Pick $r_1, \ldots, r_{n-1}$ independently and uniformly at random from $\mathbb{S}$. Let $C_i = Q_i(r_1, \ldots, r_{n-1})$. By induction, $\Pr[C_k = 0] \leq \frac{d-k}{|\mathbb{S}|}$.

If $C_k \neq 0$, then $q(x) = \sum_{i=1}^{k} C_i x^i = Q(r_1, \ldots, r_{n-1}, x) \neq 0$ has degree k, so for uniformly random $r_n \in \mathbb{S}$, $\Pr[q(r_n) = 0 \mid C_k \neq 0] \leq \frac{k}{|\mathbb{S}|}$. Finally,

$$\begin{aligned} \Pr[Q(r_1, \ldots, r_n) = 0] &\leq \Pr[C_k = 0] + \Pr[q(r_n) = 0 \mid C_k \neq 0] \\ &\leq \frac{d-k}{|\mathbb{S}|} + \frac{k}{|\mathbb{S}|} = \frac{d}{|\mathbb{S}|} \end{aligned}$$

□

You can construct Q_i simply by grouping all terms containing x_n^i and moving x_n^i outside parentheses.

Schwartz-Zippel Theorem

Proof by induction on n.

We have already proven $n = 1$. Assume $n \geq 2$ and that it holds for all smaller n. Let $Q \neq 0$, and let $k > 0$ be max exponent of x_n, then there exists $Q_0, \ldots, Q_k$ such that

$$Q(x_1, \ldots, x_n) = \sum_{i=0}^{k} Q_i(x_1, \ldots, x_{n-1}) x_n^i$$

By construction, $Q_k \neq 0$ and $\deg(Q_k) \leq d - k$. Pick $r_1, \ldots, r_{n-1}$ independently and uniformly at random from $\mathbb{S}$. Let $C_i = Q_i(r_1, \ldots, r_{n-1})$. By induction, $\Pr[C_k = 0] \leq \frac{d-k}{|\mathbb{S}|}$.

If $C_k \neq 0$, then $q(x) = \sum_{i=1}^{k} C_i x^i = Q(r_1, \ldots, r_{n-1}, x) \neq 0$ has degree k, so for uniformly random $r_n \in \mathbb{S}$, $\Pr[q(r_n) = 0 \mid C_k \neq 0] \leq \frac{k}{|\mathbb{S}|}$. Finally,

$$\begin{aligned} \Pr[Q(r_1, \ldots, r_n) = 0] &\leq \Pr[C_k = 0] + \Pr[q(r_n) = 0 \mid C_k \neq 0] \\ &\leq \frac{d-k}{|\mathbb{S}|} + \frac{k}{|\mathbb{S}|} = \frac{d}{|\mathbb{S}|} \end{aligned}$$

□

$Q_k \neq 0$ because k is an exponent of x_n that appears in Q. Every term in $Q_k(x_1, \ldots, x_{n-1}) x_n^k$ is a term in Q, so $\deg(Q_k(x_1, \ldots, x_{n-1}) x_n^k) \leq d$ and therefore $\deg(Q_k) \leq d - k$. Note that this is only a statement about Q_k, not about the other Q_i for $0 \leq i < k$.

Schwartz-Zippel Theorem

Proof by induction on n.

We have already proven $n = 1$. Assume $n \geq 2$ and that it holds for all smaller n. Let $Q \neq 0$, and let $k > 0$ be max exponent of x_n, then there exists $Q_0, \ldots, Q_k$ such that

$$Q(x_1, \ldots, x_n) = \sum_{i=0}^{k} Q_i(x_1, \ldots, x_{n-1}) x_n^i$$

By construction, $Q_k \neq 0$ and $\deg(Q_k) \leq d - k$. Pick $r_1, \ldots, r_{n-1}$ independently and uniformly at random from $\mathbb{S}$.
Let $C_i = Q_i(r_1, \ldots, r_{n-1})$. By induction, $\Pr[C_k = 0] \leq \frac{d-k}{|\mathbb{S}|}$.

If $C_k \neq 0$, then $q(x) = \sum_{i=1}^{k} C_i x^i = Q(r_1, \ldots, r_{n-1}, x) \neq 0$ has degree k, so for uniformly random $r_n \in \mathbb{S}$, $\Pr[q(r_n) = 0 \mid C_k \neq 0] \leq \frac{k}{|\mathbb{S}|}$. Finally,

$$\begin{aligned} \Pr[Q(r_1, \ldots, r_n) = 0] &\leq \Pr[C_k = 0] + \Pr[q(r_n) = 0 \mid C_k \neq 0] \\ &\leq \frac{d-k}{|\mathbb{S}|} + \frac{k}{|\mathbb{S}|} = \frac{d}{|\mathbb{S}|} \end{aligned}$$

□

We are using the *principle of deferred decisions* again. Since $r_1, \ldots, r_n$ are independent, we can pretend that $r_1, \ldots, r_{n-1}$ are picked before r_n.

Schwartz-Zippel Theorem

Proof by induction on n.

We have already proven $n = 1$. Assume $n \geq 2$ and that it holds for all smaller n. Let $Q \neq 0$, and let $k > 0$ be max exponent of x_n, then there exists $Q_0, \ldots, Q_k$ such that

$$Q(x_1, \ldots, x_n) = \sum_{i=0}^{k} Q_i(x_1, \ldots, x_{n-1}) x_n^i$$

By construction, $Q_k \neq 0$ and $\deg(Q_k) \leq d - k$. Pick $r_1, \ldots, r_{n-1}$ independently and uniformly at random from $\mathbb{S}$. Let $C_i = Q_i(r_1, \ldots, r_{n-1})$. By induction, $\Pr[C_k = 0] \leq \frac{d-k}{|\mathbb{S}|}$.

If $C_k \neq 0$, then $q(x) = \sum_{i=1}^{k} C_i x^i = Q(r_1, \ldots, r_{n-1}, x) \neq 0$ has degree k, so for uniformly random $r_n \in \mathbb{S}$, $\Pr[q(r_n) = 0 \mid C_k \neq 0] \leq \frac{k}{|\mathbb{S}|}$. Finally,

$$\begin{aligned} \Pr[Q(r_1, \ldots, r_n) = 0] &\leq \Pr[C_k = 0] + \Pr[q(r_n) = 0 \mid C_k \neq 0] \\ &\leq \frac{d-k}{|\mathbb{S}|} + \frac{k}{|\mathbb{S}|} = \frac{d}{|\mathbb{S}|} \end{aligned}$$

□

Each $C_i \in \mathbb{F}$ is just a new random variable, depending on $r_1, \ldots, r_{n-1}$.

Schwartz-Zippel Theorem

Proof by induction on n.

We have already proven $n = 1$. Assume $n \geq 2$ and that it holds for all smaller n. Let $Q \neq 0$, and let $k > 0$ be max exponent of x_n, then there exists $Q_0, \ldots, Q_k$ such that

$$Q(x_1, \ldots, x_n) = \sum_{i=0}^{k} Q_i(x_1, \ldots, x_{n-1}) x_n^i$$

By construction, $Q_k \neq 0$ and $\deg(Q_k) \leq d - k$. Pick $r_1, \ldots, r_{n-1}$ independently and uniformly at random from $\mathbb{S}$. Let $C_i = Q_i(r_1, \ldots, r_{n-1})$. By induction, $\Pr[C_k = 0] \leq \frac{d-k}{|\mathbb{S}|}$.

If $C_k \neq 0$, then $q(x) = \sum_{i=1}^{k} C_i x^i = Q(r_1, \ldots, r_{n-1}, x) \neq 0$ has degree k, so for uniformly random $r_n \in \mathbb{S}$, $\Pr[q(r_n) = 0 \mid C_k \neq 0] \leq \frac{k}{|\mathbb{S}|}$. Finally,

$$\begin{aligned}\Pr[Q(r_1, \ldots, r_n) = 0] &\leq \Pr[C_k = 0] + \Pr[q(r_n) = 0 \mid C_k \neq 0] \\ &\leq \frac{d-k}{|\mathbb{S}|} + \frac{k}{|\mathbb{S}|} = \frac{d}{|\mathbb{S}|}\end{aligned}$$

□

By definition of C_k, and our observation that $Q_k \neq 0$

$$\begin{aligned}\Pr[C_k = 0] &= \Pr[Q_k(r_1, \ldots, r_{n-1}) = 0 \mid Q_k \neq 0] \\ &\leq \frac{d-k}{|\mathbb{S}|} && \text{(by induction)}\end{aligned}$$

Schwartz-Zippel Theorem

Proof by induction on n.

We have already proven $n = 1$. Assume $n \geq 2$ and that it holds for all smaller n. Let $Q \neq 0$, and let $k > 0$ be max exponent of x_n, then there exists $Q_0, \ldots, Q_k$ such that

$$Q(x_1, \ldots, x_n) = \sum_{i=0}^{k} Q_i(x_1, \ldots, x_{n-1}) x_n^i$$

By construction, $Q_k \neq 0$ and $\deg(Q_k) \leq d - k$. Pick $r_1, \ldots, r_{n-1}$ independently and uniformly at random from $\mathbb{S}$. Let $C_i = Q_i(r_1, \ldots, r_{n-1})$. By induction, $\Pr[C_k = 0] \leq \frac{d-k}{|\mathbb{S}|}$.

If $C_k \neq 0$, then $q(x) = \sum_{i=1}^{k} C_i x^i = Q(r_1, \ldots, r_{n-1}, x) \neq 0$ has degree k, so for uniformly random $r_n \in \mathbb{S}$, $\Pr[q(r_n) = 0 \mid C_k \neq 0] \leq \frac{k}{|\mathbb{S}|}$. Finally,

$$\begin{aligned} \Pr[Q(r_1, \ldots, r_n) = 0] &\leq \Pr[C_k = 0] + \Pr[q(r_n) = 0 \mid C_k \neq 0] \\ &\leq \frac{d-k}{|\mathbb{S}|} + \frac{k}{|\mathbb{S}|} = \frac{d}{|\mathbb{S}|} \end{aligned}$$

$\square$

Schwartz-Zippel Theorem

Proof by induction on n.

We have already proven $n = 1$. Assume $n \geq 2$ and that it holds for all smaller n. Let $Q \neq 0$, and let $k > 0$ be max exponent of x_n, then there exists $Q_0, \ldots, Q_k$ such that

$$Q(x_1, \ldots, x_n) = \sum_{i=0}^{k} Q_i(x_1, \ldots, x_{n-1}) x_n^i$$

By construction, $Q_k \neq 0$ and $\deg(Q_k) \leq d - k$. Pick $r_1, \ldots, r_{n-1}$ independently and uniformly at random from $\mathbb{S}$. Let $C_i = Q_i(r_1, \ldots, r_{n-1})$. By induction, $\Pr[C_k = 0] \leq \frac{d-k}{|\mathbb{S}|}$. If $C_k \neq 0$, then $q(x) = \sum_{i=1}^{k} C_i x^i = Q(r_1, \ldots, r_{n-1}, x) \neq 0$ has degree k, so for uniformly random $r_n \in \mathbb{S}$, $\Pr[q(r_n) = 0 \mid C_k \neq 0] \leq \frac{k}{|\mathbb{S}|}$. Finally,

$$\begin{aligned}\Pr[Q(r_1, \ldots, r_n) = 0] &\leq \Pr[C_k = 0] + \Pr[q(r_n) = 0 \mid C_k \neq 0] \\ &\leq \frac{d-k}{|\mathbb{S}|} + \frac{k}{|\mathbb{S}|} = \frac{d}{|\mathbb{S}|}\end{aligned}$$

$\square$

Again by induction (actually the univariate case from the previous theorem).

Schwartz-Zippel Theorem

Proof by induction on n.

We have already proven $n = 1$. Assume $n \geq 2$ and that it holds for all smaller n. Let $Q \neq 0$, and let $k > 0$ be max exponent of x_n, then there exists $Q_0, \ldots, Q_k$ such that

$$Q(x_1, \ldots, x_n) = \sum_{i=0}^{k} Q_i(x_1, \ldots, x_{n-1}) x_n^i$$

By construction, $Q_k \neq 0$ and $\deg(Q_k) \leq d - k$. Pick $r_1, \ldots, r_{n-1}$ independently and uniformly at random from $\mathbb{S}$. Let $C_i = Q_i(r_1, \ldots, r_{n-1})$. By induction, $\Pr[C_k = 0] \leq \frac{d-k}{|\mathbb{S}|}$. If $C_k \neq 0$, then $q(x) = \sum_{i=1}^{k} C_i x^i = Q(r_1, \ldots, r_{n-1}, x) \neq 0$ has degree k, so for uniformly random $r_n \in \mathbb{S}$, $\Pr[q(r_n) = 0 \mid C_k \neq 0] \leq \frac{k}{|\mathbb{S}|}$. Finally,

$$\Pr[Q(r_1, \ldots, r_n) = 0] \leq \Pr[C_k = 0] + \Pr[q(r_n) = 0 \mid C_k \neq 0]$$

$$\leq \frac{d-k}{|\mathbb{S}|} + \frac{k}{|\mathbb{S}|} = \frac{d}{|\mathbb{S}|} \qquad \square$$

This is using Exercise 7.3 in the book:
Prove that for any two events $\mathcal{E}_1, \mathcal{E}_2$,

$$\Pr[\mathcal{E}_1] \leq \Pr[\mathcal{E}_1 \mid \overline{\mathcal{E}}_2] + \Pr[\mathcal{E}_2]$$

You should try solving this before reading the solution here.
For any events $\mathcal{E}_1, \mathcal{E}_2$, if $\Pr[\mathcal{E}_2] = 1$ then trivially

$$\Pr[\mathcal{E}_1] \leq 1 \leq \Pr[\mathcal{E}_1 \mid \overline{\mathcal{E}}_2] + 1 = \Pr[\mathcal{E}_1 \mid \overline{\mathcal{E}}_2] + \Pr[\mathcal{E}_2]$$

Otherwise $\Pr[\overline{\mathcal{E}}_2] = 1 - \Pr[\mathcal{E}_2] > 0$ and

$$\begin{aligned}
\Pr[\mathcal{E}_1] &\leq \Pr[\mathcal{E}_1 \cup \mathcal{E}_2] \\
&= \Pr[\mathcal{E}_1 \cap \overline{\mathcal{E}}_2] + \Pr[\mathcal{E}_2] \\
&\leq \frac{\Pr[\mathcal{E}_1 \cap \overline{\mathcal{E}}_2]}{\Pr[\overline{\mathcal{E}}_2]} + \Pr[\mathcal{E}_2] \\
&= \Pr[\mathcal{E}_1 \mid \overline{\mathcal{E}}_2] + \Pr[\mathcal{E}_2]
\end{aligned}$$

Schwartz-Zippel Theorem

Proof by induction on n.

We have already proven $n = 1$. Assume $n \geq 2$ and that it holds for all smaller n. Let $Q \neq 0$, and let $k > 0$ be max exponent of x_n, then there exists $Q_0, \ldots, Q_k$ such that

$$Q(x_1, \ldots, x_n) = \sum_{i=0}^{k} Q_i(x_1, \ldots, x_{n-1}) x_n^i$$

By construction, $Q_k \neq 0$ and $\deg(Q_k) \leq d - k$. Pick $r_1, \ldots, r_{n-1}$ independently and uniformly at random from $\mathbb{S}$. Let $C_i = Q_i(r_1, \ldots, r_{n-1})$. By induction, $\Pr[C_k = 0] \leq \frac{d-k}{|\mathbb{S}|}$. If $C_k \neq 0$, then $q(x) = \sum_{i=1}^{k} C_i x^i = Q(r_1, \ldots, r_{n-1}, x) \neq 0$ has degree k, so for uniformly random $r_n \in \mathbb{S}$, $\Pr[q(r_n) = 0 \mid C_k \neq 0] \leq \frac{k}{|\mathbb{S}|}$. Finally,

$$\begin{aligned}\Pr[Q(r_1, \ldots, r_n) = 0] &\leq \Pr[C_k = 0] + \Pr[q(r_n) = 0 \mid C_k \neq 0] \\ &\leq \frac{d-k}{|\mathbb{S}|} + \frac{k}{|\mathbb{S}|} = \frac{d}{|\mathbb{S}|}\end{aligned}$$

Schwartz-Zippel Theorem

Proof by induction on n.

We have already proven $n = 1$. Assume $n \geq 2$ and that it holds for all smaller n. Let $Q \neq 0$, and let $k > 0$ be max exponent of x_n, then there exists $Q_0, \ldots, Q_k$ such that

$$Q(x_1, \ldots, x_n) = \sum_{i=0}^{k} Q_i(x_1, \ldots, x_{n-1}) x_n^i$$

By construction, $Q_k \neq 0$ and $\deg(Q_k) \leq d - k$. Pick $r_1, \ldots, r_{n-1}$ independently and uniformly at random from $\mathbb{S}$. Let $C_i = Q_i(r_1, \ldots, r_{n-1})$. By induction, $\Pr[C_k = 0] \leq \frac{d-k}{|\mathbb{S}|}$. If $C_k \neq 0$, then $q(x) = \sum_{i=1}^{k} C_i x^i = Q(r_1, \ldots, r_{n-1}, x) \neq 0$ has degree k, so for uniformly random $r_n \in \mathbb{S}$, $\Pr[q(r_n) = 0 \mid C_k \neq 0] \leq \frac{k}{|\mathbb{S}|}$. Finally,

$$\begin{aligned} \Pr[Q(r_1, \ldots, r_n) = 0] &\leq \Pr[C_k = 0] + \Pr[q(r_n) = 0 \mid C_k \neq 0] \\ &\leq \frac{d-k}{|\mathbb{S}|} + \frac{k}{|\mathbb{S}|} = \frac{d}{|\mathbb{S}|} \end{aligned}$$

□

Application: Bipartite perfect matching

Theorem (Edmonds)

Given a bipartite graph $G = (U, V, E)$ with $U = V = \{1, \dots, n\}$. Let $\mathbf{A}$ be the $n \times n$ symbolic matrix defined by

$$\mathbf{A}_{ij} = \begin{cases} x_{ij} & \text{if } (i,j) \in E \\ 0 & \text{if } (i,j) \notin E \end{cases}$$

and let $Q(x_{11}, x_{12}, \dots, x_{nn}) = \det(\mathbf{A})$. Then G has a perfect matching if and only if $Q \neq 0$

Thus, by testing if the polynomial Q is nonzero, we can find out if G has a perfect matching.

Application: Bipartite perfect matching

Theorem (Edmonds)

Given a bipartite graph $G = (U, V, E)$ with $U = V = \{1, \ldots, n\}$. Let $\mathbf{A}$ be the $n \times n$ symbolic matrix defined by

$$\mathbf{A}_{ij} = \begin{cases} x_{ij} & \text{if } (i,j) \in E \\ 0 & \text{if } (i,j) \notin E \end{cases}$$

and let $Q(x_{11}, x_{12}, \ldots, x_{nn}) = \det(\mathbf{A})$. Then G has a perfect matching if and only if $Q \neq 0$

Thus, by testing if the polynomial Q is nonzero, we can find out if G has a perfect matching.

Application: Bipartite perfect matching

Proof.

Let $\mathcal{S}_n$ denote the set of all permutations of $\{1, \ldots, n\}$, and for $\pi \in \mathcal{S}_n$ let $\operatorname{sgn}(\pi) \in \{-1, 1\}$ denote the *sign* of π (the details are not important here). The determinant is defined as

$$\det(\mathbf{A}) = \sum_{\pi \in \mathcal{S}_n} \operatorname{sgn}(\pi) \prod_{i=1}^{n} \mathbf{A}_{i\pi(i)}$$

Since each x_{ij} occurs only once in $\mathbf{A}$, all terms use different variables, so there is no cancellation. Thus $Q = \det(\mathbf{A}) \neq 0$ if and only if there exists a permutation $\pi \in \mathcal{S}_n$ such that $\prod_{i=1}^{n} \mathbf{A}_{i\pi(i)} \neq 0$. This is equivalent to saying that $(i, \pi(i)) \in E$ for all $i \in \{1, \ldots, n\}$, or in other words that G contains the perfect matching corresponding to π. $\square$

Application: Bipartite perfect matching

Proof.

Let $\mathcal{S}_n$ denote the set of all permutations of $\{1, \ldots, n\}$, and for $\pi \in \mathcal{S}_n$ let $\mathrm{sgn}(\pi) \in \{-1, 1\}$ denote the *sign* of π (the details are not important here). The determinant is defined as

$$\det(\mathbf{A}) = \sum_{\pi \in \mathcal{S}_n} \mathrm{sgn}(\pi) \prod_{i=1}^{n} \mathbf{A}_{i\pi(i)}$$

Since each x_{ij} occurs only once in $\mathbf{A}$, all terms use different variables, so there is no cancellation. Thus $Q = \det(\mathbf{A}) \neq 0$ if and only if there exists a permutation $\pi \in \mathcal{S}_n$ such that $\prod_{i=1}^{n} \mathbf{A}_{i\pi(i)} \neq 0$. This is equivalent to saying that $(i, \pi(i)) \in E$ for all $i \in \{1, \ldots, n\}$, or in other words that G contains the perfect matching corresponding to π. $\square$

Application: Bipartite perfect matching

Proof.

Let $\mathcal{S}_n$ denote the set of all permutations of $\{1, \ldots, n\}$, and for $\pi \in \mathcal{S}_n$ let $\mathrm{sgn}(\pi) \in \{-1, 1\}$ denote the *sign* of π (the details are not important here). The determinant is defined as

$$\det(\mathbf{A}) = \sum_{\pi \in \mathcal{S}_n} \mathrm{sgn}(\pi) \prod_{i=1}^{n} \mathbf{A}_{i\pi(i)}$$

Since each x_{ij} occurs only once in $\mathbf{A}$, all terms use different variables, so there is no cancellation. Thus $Q = \det(\mathbf{A}) \neq 0$ if and only if there exists a permutation $\pi \in \mathcal{S}_n$ such that $\prod_{i=1}^{n} \mathbf{A}_{i\pi(i)} \neq 0$. This is equivalent to saying that $(i, \pi(i)) \in E$ for all $i \in \{1, \ldots, n\}$, or in other words that G contains the perfect matching corresponding to π. $\square$

Application: Bipartite perfect matching

Proof.

Let $\mathcal{S}_n$ denote the set of all permutations of $\{1, \ldots, n\}$, and for $\pi \in \mathcal{S}_n$ let $\mathrm{sgn}(\pi) \in \{-1, 1\}$ denote the *sign* of π (the details are not important here). The determinant is defined as

$$\det(\mathbf{A}) = \sum_{\pi \in \mathcal{S}_n} \mathrm{sgn}(\pi) \prod_{i=1}^{n} \mathbf{A}_{i\pi(i)}$$

Since each x_{ij} occurs only once in $\mathbf{A}$, all terms use different variables, so there is no cancellation. Thus $Q = \det(\mathbf{A}) \neq 0$ if and only if there exists a permutation $\pi \in \mathcal{S}_n$ such that $\prod_{i=1}^{n} \mathbf{A}_{i\pi(i)} \neq 0$. This is equivalent to saying that $(i, \pi(i)) \in E$ for all $i \in \{1, \ldots, n\}$, or in other words that G contains the perfect matching corresponding to π. $\square$

Application: Bipartite perfect matching

Proof.

Let $\mathcal{S}_n$ denote the set of all permutations of $\{1, \ldots, n\}$, and for $\pi \in \mathcal{S}_n$ let $\mathrm{sgn}(\pi) \in \{-1, 1\}$ denote the *sign* of π (the details are not important here). The determinant is defined as

$$\det(\mathbf{A}) = \sum_{\pi \in \mathcal{S}_n} \mathrm{sgn}(\pi) \prod_{i=1}^{n} \mathbf{A}_{i\pi(i)}$$

Since each x_{ij} occurs only once in $\mathbf{A}$, all terms use different variables, so there is no cancellation. Thus $Q = \det(\mathbf{A}) \neq 0$ if and only if there exists a permutation $\pi \in \mathcal{S}_n$ such that $\prod_{i=1}^{n} \mathbf{A}_{i\pi(i)} \neq 0$. This is equivalent to saying that $(i, \pi(i)) \in E$ for all $i \in \{1, \ldots, n\}$, or in other words that G contains the perfect matching corresponding to π. $\square$

No cancellation means that

$$Q = 0 \iff \forall \pi \in \mathcal{S}_n : \prod_{i=1}^{n} \mathbf{A}_{i\pi(i)} = 0$$

Or equivalently

$$Q \neq 0 \iff \exists \pi \in \mathcal{S}_n : \prod_{i=1}^{n} \mathbf{A}_{i\pi(i)} \neq 0$$

Application: Bipartite perfect matching

Proof.

Let $\mathcal{S}_n$ denote the set of all permutations of $\{1, \ldots, n\}$, and for $\pi \in \mathcal{S}_n$ let $\mathrm{sgn}(\pi) \in \{-1, 1\}$ denote the *sign* of π (the details are not important here). The determinant is defined as

$$\det(\mathbf{A}) = \sum_{\pi \in \mathcal{S}_n} \mathrm{sgn}(\pi) \prod_{i=1}^{n} \mathbf{A}_{i\pi(i)}$$

Since each x_{ij} occurs only once in $\mathbf{A}$, all terms use different variables, so there is no cancellation. Thus $Q = \det(\mathbf{A}) \neq 0$ if and only if there exists a permutation $\pi \in \mathcal{S}_n$ such that $\prod_{i=1}^{n} \mathbf{A}_{i\pi(i)} \neq 0$. This is equivalent to saying that $(i, \pi(i)) \in E$ for all $i \in \{1, \ldots, n\}$, or in other words that G contains the perfect matching corresponding to π. $\square$

Application: Bipartite perfect matching

Proof.

Let $\mathcal{S}_n$ denote the set of all permutations of $\{1, \ldots, n\}$, and for $\pi \in \mathcal{S}_n$ let $\operatorname{sgn}(\pi) \in \{-1, 1\}$ denote the *sign* of π (the details are not important here). The determinant is defined as

$$\det(\mathbf{A}) = \sum_{\pi \in \mathcal{S}_n} \operatorname{sgn}(\pi) \prod_{i=1}^{n} \mathbf{A}_{i\pi(i)}$$

Since each x_{ij} occurs only once in $\mathbf{A}$, all terms use different variables, so there is no cancellation. Thus $Q = \det(\mathbf{A}) \neq 0$ if and only if there exists a permutation $\pi \in \mathcal{S}_n$ such that $\prod_{i=1}^{n} \mathbf{A}_{i\pi(i)} \neq 0$. This is equivalent to saying that $(i, \pi(i)) \in E$ for all $i \in \{1, \ldots, n\}$, or in other words that G contains the perfect matching corresponding to π. $\square$

String equality I: (mod p) fingerprint

Suppose Alice has an n-bit string $\mathbf{a} \in \{0,1\}^n$ and Bob has an n-bit string $\mathbf{b} \in \{0,1\}^n$. They want to verify that $\mathbf{a} = \mathbf{b}$ with high probability by communicating only few (much less than n) bits. Let

$$a = \sum_{i=1}^{n} \mathbf{a}_i 2^{i-1} \qquad\qquad b = \sum_{i=1}^{n} \mathbf{b}_i 2^{i-1}$$

Theorem

For $t \geq 1$, let $\tau = \max\{\lceil tn \ln(tn)\rceil, 11\}$ and pick prime $p \leq \tau$ uniformly at random and define $F_p(x) = x \bmod p$. Then

$$\Pr[F_p(a) = F_p(b) \mid a \neq b] \in \mathcal{O}(\tfrac{1}{t})$$

So if Alice sends $p, F_p(a)$ to Bob at a cost of $\mathcal{O}(\log t + \log n)$ bits, Bob gets probability $\mathcal{O}(\frac{1}{t})$ of a false positive. And choosing $t = n$ gives cost $\mathcal{O}(\log n)$ and error rate $\mathcal{O}(\frac{1}{n})$.

String equality I: (mod p) fingerprint

Suppose Alice has an n-bit string $\mathbf{a} \in \{0,1\}^n$ and Bob has an n-bit string $\mathbf{b} \in \{0,1\}^n$. They want to verify that $\mathbf{a} = \mathbf{b}$ with high probability by communicating only few (much less than n) bits. Let

$$a = \sum_{i=1}^{n} \mathbf{a}_i 2^{i-1} \qquad b = \sum_{i=1}^{n} \mathbf{b}_i 2^{i-1}$$

Theorem

For $t \geq 1$, let $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$ and pick prime $p \leq \tau$ uniformly at random and define $F_p(x) = x \bmod p$. Then

$$\Pr[F_p(a) = F_p(b) \mid a \neq b] \in \mathcal{O}(\tfrac{1}{t})$$

So if Alice sends $p, F_p(a)$ to Bob at a cost of $\mathcal{O}(\log t + \log n)$ bits, Bob gets probability $\mathcal{O}(\frac{1}{t})$ of a false positive. And choosing $t = n$ gives cost $\mathcal{O}(\log n)$ and error rate $\mathcal{O}(\frac{1}{n})$.

String equality I: (mod p) fingerprint

Suppose Alice has an n-bit string $\mathbf{a} \in \{0,1\}^n$ and Bob has an n-bit string $\mathbf{b} \in \{0,1\}^n$. They want to verify that $\mathbf{a} = \mathbf{b}$ with high probability by communicating only few (much less than n) bits. Let

$$a = \sum_{i=1}^{n} \mathbf{a}_i 2^{i-1} \qquad\qquad b = \sum_{i=1}^{n} \mathbf{b}_i 2^{i-1}$$

Theorem

For $t \geq 1$, let $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$ and pick prime $p \leq \tau$ uniformly at random and define $F_p(x) = x \bmod p$. Then

$$\Pr[F_p(a) = F_p(b) \mid a \neq b] \in \mathcal{O}(\tfrac{1}{t})$$

So if Alice sends $p, F_p(a)$ to Bob at a cost of $\mathcal{O}(\log t + \log n)$ bits, Bob gets probability $\mathcal{O}(\frac{1}{t})$ of a false positive. And choosing $t = n$ gives cost $\mathcal{O}(\log n)$ and error rate $\mathcal{O}(\frac{1}{n})$.

String equality I: (mod p) fingerprint

Suppose Alice has an n-bit string $\mathbf{a} \in \{0,1\}^n$ and Bob has an n-bit string $\mathbf{b} \in \{0,1\}^n$. They want to verify that $\mathbf{a} = \mathbf{b}$ with high probability by communicating only few (much less than n) bits. Let

$$a = \sum_{i=1}^{n} \mathbf{a}_i 2^{i-1} \qquad\qquad b = \sum_{i=1}^{n} \mathbf{b}_i 2^{i-1}$$

Theorem

For $t \geq 1$, let $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$ and pick prime $p \leq \tau$ uniformly at random and define $F_p(x) = x \bmod p$. Then

$$\Pr[F_p(a) = F_p(b) \mid a \neq b] \in \mathcal{O}(\tfrac{1}{t})$$

So if Alice sends $p, F_p(a)$ to Bob at a cost of $\mathcal{O}(\log t + \log n)$ bits, Bob gets probability $\mathcal{O}(\frac{1}{t})$ of a false positive. And choosing $t = n$ gives cost $\mathcal{O}(\log n)$ and error rate $\mathcal{O}(\frac{1}{n})$.

String equality I: (mod p) fingerprint

Suppose Alice has an n-bit string $\mathbf{a} \in \{0,1\}^n$ and Bob has an n-bit string $\mathbf{b} \in \{0,1\}^n$. They want to verify that $\mathbf{a} = \mathbf{b}$ with high probability by communicating only few (much less than n) bits. Let

$$a = \sum_{i=1}^{n} \mathbf{a}_i 2^{i-1} \qquad\qquad b = \sum_{i=1}^{n} \mathbf{b}_i 2^{i-1}$$

Theorem

For $t \geq 1$, let $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$ and pick prime $p \leq \tau$ uniformly at random and define $F_p(x) = x \bmod p$. Then

$$\Pr[F_p(a) = F_p(b) \mid a \neq b] \in \mathcal{O}(\tfrac{1}{t})$$

So if Alice sends $p, F_p(a)$ to Bob at a cost of $\mathcal{O}(\log t + \log n)$ bits, Bob gets probability $\mathcal{O}(\frac{1}{t})$ of a false positive. And choosing $t = n$ gives cost $\mathcal{O}(\log n)$ and error rate $\mathcal{O}(\frac{1}{n})$.

String equality I: (mod p) fingerprint

Suppose Alice has an n-bit string $\mathbf{a} \in \{0,1\}^n$ and Bob has an n-bit string $\mathbf{b} \in \{0,1\}^n$. They want to verify that $\mathbf{a} = \mathbf{b}$ with high probability by communicating only few (much less than n) bits. Let

$$a = \sum_{i=1}^{n} \mathbf{a}_i 2^{i-1} \qquad\qquad b = \sum_{i=1}^{n} \mathbf{b}_i 2^{i-1}$$

Theorem

For $t \geq 1$, let $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$ and pick prime $p \leq \tau$ uniformly at random and define $F_p(x) = x \bmod p$. Then

$$\Pr[F_p(a) = F_p(b) \mid a \neq b] \in \mathcal{O}(\tfrac{1}{t})$$

So if Alice sends $p, F_p(a)$ to Bob at a cost of $\mathcal{O}(\log t + \log n)$ bits, Bob gets probability $\mathcal{O}(\frac{1}{t})$ of a false positive. And choosing $t = n$ gives cost $\mathcal{O}(\log n)$ and error rate $\mathcal{O}(\frac{1}{n})$.

Choosing e.g. $t = n^c$ gives error probability $\mathcal{O}(n^{-c})$ and uses only $\mathcal{O}((c+1)\log n)$ bits.
In other words, we have *very high* probability $1 - \mathcal{O}(n^{-c})$ of success.

String equality I: (mod p) fingerprint

Proof.

Suppose $a \neq b$, then

$$\begin{aligned} F_p(a) = F_p(b) &\iff a \bmod p = b \bmod p \\ &\iff a - b = 0 \pmod p \\ &\iff p \mid (a-b) \end{aligned}$$

Let $c = |a-b| < 2^n$. Since every prime is ≥ 2, c has at most n distinct prime divisors. Let $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$, then the number of primes $\leq \tau$ is $\pi(\tau) \geq \frac{\tau}{\ln \tau} \geq \frac{e}{e+1} tn$. Thus $\Pr\left[p \mid c \;\middle|\; a \neq b\right] \leq \frac{n}{\pi(\tau)} \leq \frac{n}{\frac{e}{e+1} tn} = (1 + \frac{1}{e}) \frac{1}{t} \in \mathcal{O}(\frac{1}{t})$. $\square$

String equality I: (mod p) fingerprint

Proof.

Suppose $a \neq b$, then

$$
\begin{aligned}
F_p(a) = F_p(b) &\iff a \bmod p = b \bmod p \\
&\iff a - b = 0 \pmod{p} \\
&\iff p \mid (a - b)
\end{aligned}
$$

Let $c = |a - b| < 2^n$. Since every prime is ≥ 2, c has at most n distinct prime divisors. Let $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$, then the number of primes $\leq \tau$ is $\pi(\tau) \geq \frac{\tau}{\ln \tau} \geq \frac{e}{e+1} tn$. Thus $\Pr\left[p \,|\, c \,\middle|\, a \neq b\right] \leq \frac{n}{\pi(\tau)} \leq \frac{n}{\frac{e}{e+1} tn} = (1 + \frac{1}{e}) \frac{1}{t} \in \mathcal{O}(\frac{1}{t})$. $\square$

String equality I: $\pmod p$ fingerprint

Proof.

Suppose $a \neq b$, then

$$\begin{aligned} F_p(a) = F_p(b) &\iff a \bmod p = b \bmod p \\ &\iff a - b = 0 \pmod p \\ &\iff p \mid (a - b) \end{aligned}$$

Let $c = |a - b| < 2^n$. Since every prime is ≥ 2, c has at most n distinct prime divisors. Let $\tau = \max\{\lceil tn\ln(tn)\rceil, 11\}$, then the number of primes $\leq \tau$ is $\pi(\tau) \geq \frac{\tau}{\ln\tau} \geq \frac{e}{e+1}tn$. Thus $\Pr[p \mid c \mid a \neq b] \leq \frac{n}{\pi(\tau)} \leq \frac{n}{\frac{e}{e+1}tn} = (1+\frac{1}{e})\frac{1}{t} \in \mathcal{O}(\frac{1}{t})$. $\square$

String equality I: (mod p) fingerprint

Proof.

Suppose $a \neq b$, then

$$\begin{aligned} F_p(a) = F_p(b) &\iff a \bmod p = b \bmod p \\ &\iff a - b = 0 \pmod{p} \\ &\iff p \mid (a - b) \end{aligned}$$

Let $c = |a - b| < 2^n$. Since every prime is ≥ 2, c has at most n distinct prime divisors. Let $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$, then the number of primes $\leq \tau$ is $\pi(\tau) \geq \frac{\tau}{\ln \tau} \geq \frac{e}{e+1} tn$. Thus $\Pr\left[p \,|\, c \;\middle|\; a \neq b\right] \leq \frac{n}{\pi(\tau)} \leq \frac{n}{\frac{e}{e+1} tn} = (1 + \frac{1}{e}) \frac{1}{t} \in \mathcal{O}(\frac{1}{t})$. $\square$

String equality I: (mod p) fingerprint

Proof.

Suppose $a \neq b$, then

$$\begin{aligned} F_p(a) = F_p(b) &\iff a \bmod p = b \bmod p \\ &\iff a - b = 0 \pmod p \\ &\iff p \mid (a - b) \end{aligned}$$

Let $c = |a - b| < 2^n$. Since every prime is ≥ 2, c has at most n distinct prime divisors. Let $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$, then the number of primes $\leq \tau$ is $\pi(\tau) \geq \frac{\tau}{\ln \tau} \geq \frac{e}{e+1} tn$. Thus $\Pr\left[p \mid c \;\middle|\; a \neq b\right] \leq \frac{n}{\pi(\tau)} \leq \frac{n}{\frac{e}{e+1} tn} = (1 + \frac{1}{e})\frac{1}{t} \in \mathcal{O}(\frac{1}{t})$.

$\square$

String equality I: (mod p) fingerprint

Proof.

Suppose $a \neq b$, then

$$\begin{aligned} F_p(a) = F_p(b) &\iff a \bmod p = b \bmod p \\ &\iff a - b = 0 \pmod p \\ &\iff p \mid (a - b) \end{aligned}$$

Let $c = |a - b| < 2^n$. Since every prime is ≥ 2, c has at most n distinct prime divisors. Let $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$, then the number of primes $\leq \tau$ is $\pi(\tau) \geq \frac{\tau}{\ln \tau} \geq \frac{e}{e+1} tn$. Thus $\Pr[p \mid c \mid a \neq b] \leq \frac{n}{\pi(\tau)} \leq \frac{n}{\frac{e}{e+1} tn} = (1 + \frac{1}{e}) \frac{1}{t} \in \mathcal{O}(\frac{1}{t})$. $\square$

String equality I: (mod p) fingerprint

Proof.

Suppose $a \neq b$, then

$$\begin{aligned} F_p(a) = F_p(b) &\iff a \bmod p = b \bmod p \\ &\iff a - b = 0 \pmod p \\ &\iff p \mid (a - b) \end{aligned}$$

Let $c = |a - b| < 2^n$. Since every prime is ≥ 2, c has at most n distinct prime divisors. Let $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$, then the number of primes $\leq \tau$ is $\pi(\tau) \geq \frac{\tau}{\ln \tau} \geq \frac{e}{e+1} tn$. Thus $\Pr\left[p \,|\, c \;\middle|\; a \neq b\right] \leq \frac{n}{\pi(\tau)} \leq \frac{n}{\frac{e}{e+1} tn} = (1 + \frac{1}{e}) \frac{1}{t} \in \mathcal{O}(\frac{1}{t})$. $\square$

String equality I: (mod p) fingerprint

Proof.

Suppose $a \neq b$, then

$$\begin{aligned} F_p(a) = F_p(b) &\iff a \bmod p = b \bmod p \\ &\iff a - b = 0 \pmod{p} \\ &\iff p \mid (a - b) \end{aligned}$$

Let $c = |a - b| < 2^n$. Since every prime is ≥ 2, c has at most n distinct prime divisors. Let $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$, then the number of primes $\leq \tau$ is $\pi(\tau) \geq \frac{\tau}{\ln \tau} \geq \frac{e}{e+1} tn$. Thus $\Pr[p | c \mid a \neq b] \leq \frac{n}{\pi(\tau)} \leq \frac{n}{\frac{e}{e+1} tn} = (1 + \frac{1}{e})\frac{1}{t} \in \mathcal{O}(\frac{1}{t})$. □

$\pi(x)$ is the *prime counting function*, defined for all $x > 0$ to be the number of primes $\leq x$.
The prime number theorem says that asymptotically, $\pi(\tau) \sim \frac{\tau}{\ln \tau}$.
In fact, for all $\tau \geq 17$ and all integer $\tau \geq 11$ it holds that $\pi(\tau) \geq \frac{\tau}{\ln \tau}$.
This is the reason for making sure that τ is at least 11.

String equality I: (mod p) fingerprint

Proof.

Suppose $a \neq b$, then

$$\begin{aligned} F_p(a) = F_p(b) &\iff a \bmod p = b \bmod p \\ &\iff a - b = 0 \pmod p \\ &\iff p \mid (a - b) \end{aligned}$$

Let $c = |a - b| < 2^n$. Since every prime is ≥ 2, c has at most n distinct prime divisors. Let $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$, then the number of primes $\leq \tau$ is $\pi(\tau) \geq \frac{\tau}{\ln \tau} \geq \frac{e}{e+1} tn$. Thus $\Pr\left[p \,|\, c \;\middle|\; a \neq b\right] \leq \frac{n}{\pi(\tau)} \leq \frac{n}{\frac{e}{e+1} tn} = (1 + \frac{1}{e})\frac{1}{t} \in \mathcal{O}(\frac{1}{t})$.

□

For $x \ln x < 11$ we have $x < 6.08911$ so $\frac{e}{e+1} x \leq 4.4515 < 4.5873563056667095 \approx \frac{11}{\ln 11}$. So if $tn \ln(tn) \leq 11$ we have $\tau = 11$ and $\frac{\tau}{\ln \tau} \geq \frac{e}{e+1} tn$.

For $x \ln x \geq e$, we have $\frac{\lceil x \ln x \rceil}{\ln \lceil x \ln x \rceil} \geq \frac{x \ln x}{\ln(x \ln x)} \geq \frac{e}{e+1} x$. So if $tn \ln(tn) > 11 > e$, we have $\tau = \lceil tn \ln(tn) \rceil$ and again $\frac{\tau}{\ln \tau} \geq \frac{e}{e+1} tn$.

String equality I: (mod p) fingerprint

Proof.

Suppose $a \neq b$, then

$$\begin{aligned} F_p(a) = F_p(b) &\iff a \bmod p = b \bmod p \\ &\iff a - b = 0 \pmod{p} \\ &\iff p \mid (a - b) \end{aligned}$$

Let $c = |a - b| < 2^n$. Since every prime is ≥ 2, c has at most n distinct prime divisors. Let $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$, then the number of primes $\leq \tau$ is $\pi(\tau) \geq \frac{\tau}{\ln \tau} \geq \frac{e}{e+1} tn$. Thus $\Pr\left[p \mid c \;\middle|\; a \neq b\right] \leq \frac{n}{\pi(\tau)} \leq \frac{n}{\frac{e}{e+1} tn} = (1 + \frac{1}{e}) \frac{1}{t} \in \mathcal{O}(\frac{1}{t})$. $\square$

String equality I: (mod p) fingerprint

Proof.

Suppose $a \neq b$, then

$$\begin{aligned} F_p(a) = F_p(b) &\iff a \bmod p = b \bmod p \\ &\iff a - b = 0 \pmod p \\ &\iff p \mid (a - b) \end{aligned}$$

Let $c = |a - b| < 2^n$. Since every prime is ≥ 2, c has at most n distinct prime divisors. Let $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$, then the number of primes $\leq \tau$ is $\pi(\tau) \geq \frac{\tau}{\ln \tau} \geq \frac{e}{e+1} tn$. Thus $\Pr\left[p \mid c \;\middle|\; a \neq b\right] \leq \frac{n}{\pi(\tau)} \leq \frac{n}{\frac{e}{e+1} tn} = (1 + \frac{1}{e}) \frac{1}{t} \in \mathcal{O}(\frac{1}{t})$.

□

String equality I: (mod p) fingerprint

Proof.

Suppose $a \neq b$, then

$$\begin{aligned} F_p(a) = F_p(b) &\iff a \bmod p = b \bmod p \\ &\iff a - b = 0 \pmod p \\ &\iff p \mid (a - b) \end{aligned}$$

Let $c = |a - b| < 2^n$. Since every prime is ≥ 2, c has at most n distinct prime divisors. Let $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$, then the number of primes $\leq \tau$ is $\pi(\tau) \geq \frac{\tau}{\ln \tau} \geq \frac{e}{e+1} tn$. Thus $\Pr\left[p \mid c \;\middle|\; a \neq b\right] \leq \frac{n}{\pi(\tau)} \leq \frac{n}{\frac{e}{e+1} tn} = (1 + \frac{1}{e})\frac{1}{t} \in \mathcal{O}(\frac{1}{t})$. $\square$

String equality I: (mod p) fingerprint

Proof.

Suppose $a \neq b$, then

$$\begin{aligned} F_p(a) = F_p(b) &\iff a \bmod p = b \bmod p \\ &\iff a - b = 0 \pmod p \\ &\iff p \mid (a - b) \end{aligned}$$

Let $c = |a - b| < 2^n$. Since every prime is ≥ 2, c has at most n distinct prime divisors. Let $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$, then the number of primes $\leq \tau$ is $\pi(\tau) \geq \frac{\tau}{\ln \tau} \geq \frac{e}{e+1} tn$. Thus $\Pr\left[p \,|\, c \;\middle|\; a \neq b\right] \leq \frac{n}{\pi(\tau)} \leq \frac{n}{\frac{e}{e+1} tn} = (1 + \frac{1}{e}) \frac{1}{t} \in \mathcal{O}(\frac{1}{t})$. $\square$

String equality II: Freivald

Suppose Alice has an n-bit string $\mathbf{a} \in \{0,1\}^n$ and Bob has an n-bit string $\mathbf{b} \in \{0,1\}^n$. They want to verify that $\mathbf{a} = \mathbf{b}$ with high probability by communicating only few (much less than n) bits. For $t \geq 1$, choose prime $p > n$, $p \in \Theta(tn)$ and let

$$A(z) = \sum_{i=1}^{n} \mathbf{a}_i z^{i-1} \in \mathbb{Z}_p[z] \qquad B(z) = \sum_{i=1}^{n} \mathbf{b}_i z^{i-1} \in \mathbb{Z}_p[z]$$

Pick $r \in \mathbb{Z}_p$ uniformly at random, then by earlier theorem

$$\Pr[A(r) - B(r) = 0 \mid A \neq B] \leq \tfrac{n}{p} \in \mathcal{O}(\tfrac{n}{tn}) = \mathcal{O}(\tfrac{1}{t})$$

So if Alice sends $p, r, A(r)$ to Bob at a cost of $\mathcal{O}(\log t + \log n)$ bits, Bob gets probability $\mathcal{O}(\frac{1}{t})$ of a false positive. And choosing $t = n$ gives cost $\mathcal{O}(\log n)$ and error rate $\mathcal{O}(\frac{1}{n})$.

String equality II: Freivald

Suppose Alice has an n-bit string $\mathbf{a} \in \{0,1\}^n$ and Bob has an n-bit string $\mathbf{b} \in \{0,1\}^n$. They want to verify that $\mathbf{a} = \mathbf{b}$ with high probability by communicating only few (much less than n) bits. For $t \geq 1$, choose prime $p > n$, $p \in \Theta(tn)$ and let

$$A(z) = \sum_{i=1}^{n} \mathbf{a}_i z^{i-1} \in \mathbb{Z}_p[z] \qquad B(z) = \sum_{i=1}^{n} \mathbf{b}_i z^{i-1} \in \mathbb{Z}_p[z]$$

Pick $r \in \mathbb{Z}_p$ uniformly at random, then by earlier theorem

$$\Pr[A(r) - B(r) = 0 \mid A \neq B] \leq \tfrac{n}{p} \in \mathcal{O}(\tfrac{n}{tn}) = \mathcal{O}(\tfrac{1}{t})$$

So if Alice sends $p, r, A(r)$ to Bob at a cost of $\mathcal{O}(\log t + \log n)$ bits, Bob gets probability $\mathcal{O}(\frac{1}{t})$ of a false positive. And choosing $t = n$ gives cost $\mathcal{O}(\log n)$ and error rate $\mathcal{O}(\frac{1}{n})$.

String equality II: Freivald

Suppose Alice has an n-bit string $\mathbf{a} \in \{0,1\}^n$ and Bob has an n-bit string $\mathbf{b} \in \{0,1\}^n$. They want to verify that $\mathbf{a} = \mathbf{b}$ with high probability by communicating only few (much less than n) bits. For $t \geq 1$, choose prime $p > n, p \in \Theta(tn)$ and let

$$A(z) = \sum_{i=1}^{n} \mathbf{a}_i z^{i-1} \in \mathbb{Z}_p[z] \qquad B(z) = \sum_{i=1}^{n} \mathbf{b}_i z^{i-1} \in \mathbb{Z}_p[z]$$

Pick $r \in \mathbb{Z}_p$ uniformly at random, then by earlier theorem

$$\Pr[A(r) - B(r) = 0 \mid A \neq B] \leq \tfrac{n}{p} \in \mathcal{O}(\tfrac{n}{tn}) = \mathcal{O}(\tfrac{1}{t})$$

So if Alice sends $p, r, A(r)$ to Bob at a cost of $\mathcal{O}(\log t + \log n)$ bits, Bob gets probability $\mathcal{O}(\frac{1}{t})$ of a false positive. And choosing $t = n$ gives cost $\mathcal{O}(\log n)$ and error rate $\mathcal{O}(\frac{1}{n})$.

String equality II: Freivald

Suppose Alice has an n-bit string $\mathbf{a} \in \{0,1\}^n$ and Bob has an n-bit string $\mathbf{b} \in \{0,1\}^n$. They want to verify that $\mathbf{a} = \mathbf{b}$ with high probability by communicating only few (much less than n) bits. For $t \geq 1$, choose prime $p > n, p \in \Theta(tn)$ and let

$$A(z) = \sum_{i=1}^{n} \mathbf{a}_i z^{i-1} \in \mathbb{Z}_p[z] \qquad B(z) = \sum_{i=1}^{n} \mathbf{b}_i z^{i-1} \in \mathbb{Z}_p[z]$$

Pick $r \in \mathbb{Z}_p$ uniformly at random, then by earlier theorem

$$\Pr[A(r) - B(r) = 0 \mid A \neq B] \leq \tfrac{n}{p} \in \mathcal{O}(\tfrac{n}{tn}) = \mathcal{O}(\tfrac{1}{t})$$

So if Alice sends $p, r, A(r)$ to Bob at a cost of $\mathcal{O}(\log t + \log n)$ bits, Bob gets probability $\mathcal{O}(\frac{1}{t})$ of a false positive. And choosing $t = n$ gives cost $\mathcal{O}(\log n)$ and error rate $\mathcal{O}(\frac{1}{n})$.

String equality II: Freivald

Suppose Alice has an n-bit string $\mathbf{a} \in \{0,1\}^n$ and Bob has an n-bit string $\mathbf{b} \in \{0,1\}^n$. They want to verify that $\mathbf{a} = \mathbf{b}$ with high probability by communicating only few (much less than n) bits. For $t \geq 1$, choose prime $p > n, p \in \Theta(tn)$ and let

$$A(z) = \sum_{i=1}^{n} \mathbf{a}_i z^{i-1} \in \mathbb{Z}_p[z] \qquad B(z) = \sum_{i=1}^{n} \mathbf{b}_i z^{i-1} \in \mathbb{Z}_p[z]$$

Pick $r \in \mathbb{Z}_p$ uniformly at random, then by earlier theorem

$$\Pr[A(r) - B(r) = 0 \mid A \neq B] \leq \tfrac{n}{p} \in \mathcal{O}(\tfrac{n}{tn}) = \mathcal{O}(\tfrac{1}{t})$$

So if Alice sends $p, r, A(r)$ to Bob at a cost of $\mathcal{O}(\log t + \log n)$ bits, Bob gets probability $\mathcal{O}(\frac{1}{t})$ of a false positive. And choosing $t = n$ gives cost $\mathcal{O}(\log n)$ and error rate $\mathcal{O}(\frac{1}{n})$.

String equality II: Freivald

Suppose Alice has an n-bit string $\mathbf{a} \in \{0,1\}^n$ and Bob has an n-bit string $\mathbf{b} \in \{0,1\}^n$. They want to verify that $\mathbf{a} = \mathbf{b}$ with high probability by communicating only few (much less than n) bits. For $t \geq 1$, choose prime $p > n, p \in \Theta(tn)$ and let

$$A(z) = \sum_{i=1}^{n} \mathbf{a}_i z^{i-1} \in \mathbb{Z}_p[z] \qquad B(z) = \sum_{i=1}^{n} \mathbf{b}_i z^{i-1} \in \mathbb{Z}_p[z]$$

Pick $r \in \mathbb{Z}_p$ uniformly at random, then by earlier theorem

$$\Pr[A(r) - B(r) = 0 \mid A \neq B] \leq \tfrac{n}{p} \in \mathcal{O}(\tfrac{n}{tn}) = \mathcal{O}(\tfrac{1}{t})$$

So if Alice sends $p, r, A(r)$ to Bob at a cost of $\mathcal{O}(\log t + \log n)$ bits, Bob gets probability $\mathcal{O}(\frac{1}{t})$ of a false positive. And choosing $t = n$ gives cost $\mathcal{O}(\log n)$ and error rate $\mathcal{O}(\frac{1}{n})$.

String equality I vs II: Comparison

The two methods are similar but different:

I: Picks random prime $p \leq \tau$ where $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$ and uses

$$F_{p,2}(\mathbf{x}) = \left(\sum_{i=1}^{n} \mathbf{x}_i 2^{i-1}\right) \bmod p$$

II: Picks deterministic prime $p \in \Theta(tn)$, $p > n$ and random $r \in \mathbb{Z}_p$ and uses

$$F_{p,r}(\mathbf{x}) = \left(\sum_{i=1}^{n} \mathbf{x}_i r^{i-1}\right) \bmod p$$

Both have fingerprint size $\mathcal{O}(\log t + \log n)$ and error rate $\mathcal{O}(\frac{1}{t})$.

String equality I vs II: Comparison

The two methods are similar but different:

I: Picks random prime $p \le \tau$ where $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$ and uses

$$F_{p,2}(\mathbf{x}) = \left(\sum_{i=1}^{n} \mathbf{x}_i 2^{i-1} \right) \bmod p$$

II: Picks deterministic prime $p \in \Theta(tn)$, $p > n$ and random $r \in \mathbb{Z}_p$ and uses

$$F_{p,r}(\mathbf{x}) = \left(\sum_{i=1}^{n} \mathbf{x}_i r^{i-1} \right) \bmod p$$

Both have fingerprint size $\mathcal{O}(\log t + \log n)$ and error rate $\mathcal{O}(\frac{1}{t})$.

String equality I vs II: Comparison

The two methods are similar but different:

I: Picks random prime $p \leq \tau$ where $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$ and uses

$$F_{p,2}(\mathbf{x}) = \left(\sum_{i=1}^{n} \mathbf{x}_i 2^{i-1} \right) \bmod p$$

II: Picks deterministic prime $p \in \Theta(tn)$, $p > n$ and random $r \in \mathbb{Z}_p$ and uses

$$F_{p,r}(\mathbf{x}) = \left(\sum_{i=1}^{n} \mathbf{x}_i r^{i-1} \right) \bmod p$$

Both have fingerprint size $\mathcal{O}(\log t + \log n)$ and error rate $\mathcal{O}(\frac{1}{t})$.

String equality I vs II: Comparison

The two methods are similar but different:

I: Picks random prime $p \le \tau$ where $\tau = \max\{\lceil tn \ln(tn) \rceil, 11\}$ and uses

$$F_{p,2}(\mathbf{x}) = \left(\sum_{i=1}^{n} \mathbf{x}_i 2^{i-1}\right) \bmod p$$

II: Picks deterministic prime $p \in \Theta(tn)$, $p > n$ and random $r \in \mathbb{Z}_p$ and uses

$$F_{p,r}(\mathbf{x}) = \left(\sum_{i=1}^{n} \mathbf{x}_i r^{i-1}\right) \bmod p$$

Both have fingerprint size $\mathcal{O}(\log t + \log n)$ and error rate $\mathcal{O}(\frac{1}{t})$.

Pattern matching

We can use the first fingerprinting method for string equality to solve a different problem.

Given bit strings $\mathbf{a} \in \{0,1\}^m$ and $\mathbf{b} \in \{0,1\}^n$ with $m \leq n$. Find min/max j (if it exists) such that

$$\mathbf{a}_1 = \mathbf{b}_{j+1} \qquad \mathbf{a}_2 = \mathbf{b}_{j+2} \qquad \dots \qquad \mathbf{a}_m = \mathbf{b}_{j+m}$$

Trivial to solve in $\mathcal{O}(mn)$ time. Deterministic $\mathcal{O}(m+n)$ exists. We'll show simple randomized $\mathcal{O}(m+n)$. First Monte Carlo, then Las Vegas in two different ways.

Pattern matching

We can use the first fingerprinting method for string equality to solve a different problem.

Given bit strings $\mathbf{a} \in \{0,1\}^m$ and $\mathbf{b} \in \{0,1\}^n$ with $m \leq n$. Find min/max j (if it exists) such that

$$\mathbf{a}_1 = \mathbf{b}_{j+1} \qquad \mathbf{a}_2 = \mathbf{b}_{j+2} \qquad \dots \qquad \mathbf{a}_m = \mathbf{b}_{j+m}$$

Trivial to solve in $\mathcal{O}(mn)$ time. Deterministic $\mathcal{O}(m+n)$ exists. We'll show simple randomized $\mathcal{O}(m+n)$. First Monte Carlo, then Las Vegas in two different ways.

Pattern matching

We can use the first fingerprinting method for string equality to solve a different problem.

Given bit strings $\mathbf{a} \in \{0,1\}^m$ and $\mathbf{b} \in \{0,1\}^n$ with $m \leq n$. Find min/max j (if it exists) such that

$$\mathbf{a}_1 = \mathbf{b}_{j+1} \qquad \mathbf{a}_2 = \mathbf{b}_{j+2} \qquad \dots \qquad \mathbf{a}_m = \mathbf{b}_{j+m}$$

Trivial to solve in $\mathcal{O}(mn)$ time. Deterministic $\mathcal{O}(m+n)$ exists. We'll show simple randomized $\mathcal{O}(m+n)$. First Monte Carlo, then Las Vegas in two different ways.

Pattern matching

We can use the first fingerprinting method for string equality to solve a different problem.

Given bit strings $\mathbf{a} \in \{0,1\}^m$ and $\mathbf{b} \in \{0,1\}^n$ with $m \leq n$. Find min/max j (if it exists) such that

$$\mathbf{a}_1 = \mathbf{b}_{j+1} \qquad \mathbf{a}_2 = \mathbf{b}_{j+2} \qquad \ldots \qquad \mathbf{a}_m = \mathbf{b}_{j+m}$$

Trivial to solve in $\mathcal{O}(mn)$ time. Deterministic $\mathcal{O}(m+n)$ exists. We'll show simple randomized $\mathcal{O}(m+n)$. First Monte Carlo, then Las Vegas in two different ways.

Pattern matching

We can use the first fingerprinting method for string equality to solve a different problem.

Given bit strings $\mathbf{a} \in \{0,1\}^m$ and $\mathbf{b} \in \{0,1\}^n$ with $m \leq n$. Find min/max j (if it exists) such that

$$\mathbf{a}_1 = \mathbf{b}_{j+1} \qquad \mathbf{a}_2 = \mathbf{b}_{j+2} \qquad \ldots \qquad \mathbf{a}_m = \mathbf{b}_{j+m}$$

Trivial to solve in $\mathcal{O}(mn)$ time. Deterministic $\mathcal{O}(m+n)$ exists. We'll show simple randomized $\mathcal{O}(m+n)$. First Monte Carlo, then Las Vegas in two different ways.

Pattern matching

We can use the first fingerprinting method for string equality to solve a different problem.

Given bit strings $\mathbf{a} \in \{0,1\}^m$ and $\mathbf{b} \in \{0,1\}^n$ with $m \leq n$. Find min/max j (if it exists) such that

$$\mathbf{a}_1 = \mathbf{b}_{j+1} \quad \mathbf{a}_2 = \mathbf{b}_{j+2} \quad \dots \quad \mathbf{a}_m = \mathbf{b}_{j+m}$$

Trivial to solve in $\mathcal{O}(mn)$ time. Deterministic $\mathcal{O}(m+n)$ exists. We'll show simple randomized $\mathcal{O}(m+n)$. First Monte Carlo, then Las Vegas in two different ways.

Pattern matching

We can use the first fingerprinting method for string equality to solve a different problem.

Given bit strings $\mathbf{a} \in \{0,1\}^m$ and $\mathbf{b} \in \{0,1\}^n$ with $m \leq n$. Find min/max j (if it exists) such that

$$\mathbf{a}_1 = \mathbf{b}_{j+1} \qquad \mathbf{a}_2 = \mathbf{b}_{j+2} \qquad \ldots \qquad \mathbf{a}_m = \mathbf{b}_{j+m}$$

Trivial to solve in $\mathcal{O}(mn)$ time. Deterministic $\mathcal{O}(m+n)$ exists. We'll show simple randomized $\mathcal{O}(m+n)$. First Monte Carlo, then Las Vegas in two different ways.

Pattern matching

We can use the first fingerprinting method for string equality to solve a different problem.

Given bit strings $\mathbf{a} \in \{0,1\}^m$ and $\mathbf{b} \in \{0,1\}^n$ with $m \leq n$. Find min/max j (if it exists) such that

$$\mathbf{a}_1 = \mathbf{b}_{j+1} \qquad \mathbf{a}_2 = \mathbf{b}_{j+2} \qquad \ldots \qquad \mathbf{a}_m = \mathbf{b}_{j+m}$$

Trivial to solve in $\mathcal{O}(mn)$ time. Deterministic $\mathcal{O}(m+n)$ exists. We'll show simple randomized $\mathcal{O}(m+n)$. First Monte Carlo, then Las Vegas in two different ways.

Pattern matching: Monte Carlo

Define $a = \sum_{i=1}^{m} \mathbf{a}_i 2^{i-1}$ and $B_j = \sum_{i=1}^{m} \mathbf{b}_{j+i} 2^{i-1}$, let $t = n^2$, $\tau = \max\{\lceil tm \ln(tm) \rceil, 11\}$ and pick uniformly random prime $p \leq \tau$.

The idea now is to compare $F_p(a)$ to $F_p(B_j)$ for $j \in [n+1-m]$. By a union bound, $\Pr[\text{false match}] \leq (n+1-m)\mathcal{O}(\frac{1}{n^2}) = \mathcal{O}(\frac{1}{n})$.

Assuming each operation in $\mathbb{Z}_p$ takes $\mathcal{O}(1)$ time, the main problem is computing the fingerprints in $\mathcal{O}(m+n)$ total time.

Pattern matching: Monte Carlo

Define $a = \sum_{i=1}^{m} \mathbf{a}_i 2^{i-1}$ and $B_j = \sum_{i=1}^{m} \mathbf{b}_{j+i} 2^{i-1}$, let $t = n^2$, $\tau = \max\{\lceil tm \ln(tm) \rceil, 11\}$ and pick uniformly random prime $p \leq \tau$.

The idea now is to compare $F_p(a)$ to $F_p(B_j)$ for $j \in [n+1-m]$. By a union bound, $\Pr[\text{false match}] \leq (n+1-m)\mathcal{O}(\frac{1}{n^2}) = \mathcal{O}(\frac{1}{n})$.

Assuming each operation in $\mathbb{Z}_p$ takes $\mathcal{O}(1)$ time, the main problem is computing the fingerprints in $\mathcal{O}(m+n)$ total time.

Pattern matching: Monte Carlo

Define $a = \sum_{i=1}^{m} \mathbf{a}_i 2^{i-1}$ and $B_j = \sum_{i=1}^{m} \mathbf{b}_{j+i} 2^{i-1}$, let $t = n^2$, $\tau = \max\{\lceil tm \ln(tm) \rceil, 11\}$ and pick uniformly random prime $p \leq \tau$.

The idea now is to compare $F_p(a)$ to $F_p(B_j)$ for $j \in [n+1-m]$. By a union bound, $\Pr[\text{false match}] \leq (n+1-m)\mathcal{O}(\frac{1}{n^2}) = \mathcal{O}(\frac{1}{n})$.

Assuming each operation in $\mathbb{Z}_p$ takes $\mathcal{O}(1)$ time, the main problem is computing the fingerprints in $\mathcal{O}(m+n)$ total time.

Pattern matching: Monte Carlo

Define $a = \sum_{i=1}^{m} \mathbf{a}_i 2^{i-1}$ and $B_j = \sum_{i=1}^{m} \mathbf{b}_{j+i} 2^{i-1}$, let $t = n^2$, $\tau = \max\{\lceil tm \ln(tm) \rceil, 11\}$ and pick uniformly random prime $p \le \tau$.

The idea now is to compare $F_p(a)$ to $F_p(B_j)$ for $j \in [n+1-m]$. By a union bound, $\Pr[\text{false match}] \le (n+1-m)\mathcal{O}(\frac{1}{n^2}) = \mathcal{O}(\frac{1}{n})$.

Assuming each operation in $\mathbb{Z}_p$ takes $\mathcal{O}(1)$ time, the main problem is computing the fingerprints in $\mathcal{O}(m+n)$ total time.

Pattern matching: Efficient Fingerprints

Observe that

$$F_p(B_j) - 2F_p(B_{j+1}) = \Big(\sum_{i=1}^{m} b_{j+i}2^{i-1} - 2\sum_{i=1}^{m} b_{j+1+i}2^{i-1}\Big) \bmod p$$

$$= \Big(\sum_{i=1}^{m} b_{j+i}2^{i-1} - \sum_{i=2}^{m+1} b_{j+i}2^{i-1}\Big) \bmod p$$

$$= (b_{j+1} - b_{j+m+1}2^m) \bmod p$$

So we can isolate either $F_p(B_j)$ or $F_p(B_{j+1})$ and get

$$F_p(B_j) = (b_{j+1} + 2F_p(B_{j+1}) - b_{j+m+1}2^m) \bmod p$$

$$F_p(B_{j+1}) = 2^{-1}(b_{j+m+1}2^m + F_p(B_j) - b_{j+1}) \bmod p$$

Thus, by precomputing $2^m \bmod p$ and $2^{-1} \bmod p$ we can start by computing either B_{n-m} or B_0 in $\mathcal{O}(m)$ time, and then compute each of the remaining $n - m$ fingerprints in $\mathcal{O}(1)$ time each. The total time is therefore $\mathcal{O}(m + n)$.

Pattern matching: Efficient Fingerprints

Observe that

$$\begin{aligned} F_p(B_j) - 2F_p(B_{j+1}) &= \Big(\sum_{i=1}^{m} b_{j+i}2^{i-1} - 2\sum_{i=1}^{m} b_{j+1+i}2^{i-1}\Big) \bmod p \\ &= \Big(\sum_{i=1}^{m} b_{j+i}2^{i-1} - \sum_{i=2}^{m+1} b_{j+i}2^{i-1}\Big) \bmod p \\ &= (b_{j+1} - b_{j+m+1}2^{m}) \bmod p \end{aligned}$$

So we can isolate either $F_p(B_j)$ or $F_p(B_{j+1})$ and get

$$F_p(B_j) = (b_{j+1} + 2F_p(B_{j+1}) - b_{j+m+1}2^{m}) \bmod p$$

$$F_p(B_{j+1}) = 2^{-1}(b_{j+m+1}2^{m} + F_p(B_j) - b_{j+1}) \bmod p$$

Thus, by precomputing $2^m \bmod p$ and $2^{-1} \bmod p$ we can start by computing either B_{n-m} or B_0 in $\mathcal{O}(m)$ time, and then compute each of the remaining $n - m$ fingerprints in $\mathcal{O}(1)$ time each. The total time is therefore $\mathcal{O}(m + n)$.

Pattern matching: Efficient Fingerprints

Observe that

$$\begin{aligned}F_p(B_j) - 2F_p(B_{j+1}) &= \left(\sum_{i=1}^{m} b_{j+i}2^{i-1} - 2\sum_{i=1}^{m} b_{j+1+i}2^{i-1}\right) \bmod p\\ &= \left(\sum_{i=1}^{m} b_{j+i}2^{i-1} - \sum_{i=2}^{m+1} b_{j+i}2^{i-1}\right) \bmod p\\ &= (b_{j+1} - b_{j+m+1}2^{m}) \bmod p\end{aligned}$$

So we can isolate either $F_p(B_j)$ or $F_p(B_{j+1})$ and get

$$F_p(B_j) = (b_{j+1} + 2F_p(B_{j+1}) - b_{j+m+1}2^m) \bmod p$$

$$F_p(B_{j+1}) = 2^{-1}(b_{j+m+1}2^m + F_p(B_j) - b_{j+1}) \bmod p$$

Thus, by precomputing $2^m \bmod p$ and $2^{-1} \bmod p$ we can start by computing either B_{n-m} or B_0 in $\mathcal{O}(m)$ time, and then compute each of the remaining $n-m$ fingerprints in $\mathcal{O}(1)$ time each. The total time is therefore $\mathcal{O}(m+n)$.

Pattern matching: Efficient Fingerprints

Observe that

$$
\begin{aligned}
F_p(B_j) - 2F_p(B_{j+1}) &= \Big(\sum_{i=1}^{m} b_{j+i}2^{i-1} - 2\sum_{i=1}^{m} b_{j+1+i}2^{i-1}\Big) \bmod p \\
&= \Big(\sum_{i=1}^{m} b_{j+i}2^{i-1} - \sum_{i=2}^{m+1} b_{j+i}2^{i-1}\Big) \bmod p \\
&= (b_{j+1} - b_{j+m+1}2^{m}) \bmod p
\end{aligned}
$$

So we can isolate either $F_p(B_j)$ or $F_p(B_{j+1})$ and get

$$F_p(B_j) = (b_{j+1} + 2F_p(B_{j+1}) - b_{j+m+1}2^{m}) \bmod p$$

$$F_p(B_{j+1}) = 2^{-1}(b_{j+m+1}2^{m} + F_p(B_j) - b_{j+1}) \bmod p$$

Thus, by precomputing $2^m \bmod p$ and $2^{-1} \bmod p$ we can start by computing either B_{n-m} or B_0 in $\mathcal{O}(m)$ time, and then compute each of the remaining $n-m$ fingerprints in $\mathcal{O}(1)$ time each. The total time is therefore $\mathcal{O}(m+n)$.

Pattern matching: Efficient Fingerprints

Observe that

$$\begin{aligned} F_p(B_j) - 2F_p(B_{j+1}) &= \Big(\sum_{i=1}^{m} b_{j+i}2^{i-1} - 2\sum_{i=1}^{m} b_{j+1+i}2^{i-1}\Big) \bmod p \\ &= \Big(\sum_{i=1}^{m} b_{j+i}2^{i-1} - \sum_{i=2}^{m+1} b_{j+i}2^{i-1}\Big) \bmod p \\ &= (b_{j+1} - b_{j+m+1}2^{m}) \bmod p \end{aligned}$$

So we can isolate either $F_p(B_j)$ or $F_p(B_{j+1})$ and get

$$F_p(B_j) = (b_{j+1} + 2F_p(B_{j+1}) - b_{j+m+1}2^{m}) \bmod p$$

$$F_p(B_{j+1}) = 2^{-1}(b_{j+m+1}2^{m} + F_p(B_j) - b_{j+1}) \bmod p$$

Thus, by precomputing $2^m \bmod p$ and $2^{-1} \bmod p$ we can start by computing either B_{n-m} or B_0 in $\mathcal{O}(m)$ time, and then compute each of the remaining $n-m$ fingerprints in $\mathcal{O}(1)$ time each. The total time is therefore $\mathcal{O}(m+n)$.

Pattern matching: Efficient Fingerprints

Observe that

$$
\begin{aligned}
F_p(B_j) - 2F_p(B_{j+1}) &= \left(\sum_{i=1}^{m} b_{j+i}2^{i-1} - 2\sum_{i=1}^{m} b_{j+1+i}2^{i-1}\right) \bmod p \\
&= \left(\sum_{i=1}^{m} b_{j+i}2^{i-1} - \sum_{i=2}^{m+1} b_{j+i}2^{i-1}\right) \bmod p \\
&= (b_{j+1} - b_{j+m+1}2^m) \bmod p
\end{aligned}
$$

So we can isolate either $F_p(B_j)$ or $F_p(B_{j+1})$ and get

$$F_p(B_j) = (b_{j+1} + 2F_p(B_{j+1}) - b_{j+m+1}2^m) \bmod p$$

$$F_p(B_{j+1}) = 2^{-1}(b_{j+m+1}2^m + F_p(B_j) - b_{j+1}) \bmod p$$

Thus, by precomputing $2^m \bmod p$ and $2^{-1} \bmod p$ we can start by computing either B_{n-m} or B_0 in $\mathcal{O}(m)$ time, and then compute each of the remaining $n-m$ fingerprints in $\mathcal{O}(1)$ time each. The total time is therefore $\mathcal{O}(m+n)$.

By Fermat's little theorem, we have $a^{p-1} \equiv 1 \pmod{p}$ for $a \not\equiv 0 \pmod{p}$.
In particular (abusing notation), $2^{-1} \equiv 2^{p-2} \pmod{p}$, so both $2^m \pmod{p}$ and $2^{-1} \pmod{p}$ can be computed, in $\mathcal{O}(\log m)$ and $\mathcal{O}(\log p) = \mathcal{O}(\log n)$ time respectively, by the method of repeated squaring.

Pattern matching: Efficient Fingerprints

Observe that

$$
\begin{aligned}
F_p(B_j) - 2F_p(B_{j+1}) &= \Big(\sum_{i=1}^{m} b_{j+i}2^{i-1} - 2\sum_{i=1}^{m} b_{j+1+i}2^{i-1}\Big) \bmod p \\
&= \Big(\sum_{i=1}^{m} b_{j+i}2^{i-1} - \sum_{i=2}^{m+1} b_{j+i}2^{i-1}\Big) \bmod p \\
&= (b_{j+1} - b_{j+m+1}2^m) \bmod p
\end{aligned}
$$

So we can isolate either $F_p(B_j)$ or $F_p(B_{j+1})$ and get

$$F_p(B_j) = (b_{j+1} + 2F_p(B_{j+1}) - b_{j+m+1}2^m) \bmod p$$

$$F_p(B_{j+1}) = 2^{-1}(b_{j+m+1}2^m + F_p(B_j) - b_{j+1}) \bmod p$$

Thus, by precomputing $2^m \bmod p$ and $2^{-1} \bmod p$ we can start by computing either B_{n-m} or B_0 in $\mathcal{O}(m)$ time, and then compute each of the remaining $n-m$ fingerprints in $\mathcal{O}(1)$ time each. The total time is therefore $\mathcal{O}(m+n)$.

Pattern matching: Efficient Fingerprints

Observe that

$$
\begin{aligned}
F_p(B_j) - 2F_p(B_{j+1}) &= \left(\sum_{i=1}^{m} b_{j+i}2^{i-1} - 2\sum_{i=1}^{m} b_{j+1+i}2^{i-1}\right) \bmod p \\
&= \left(\sum_{i=1}^{m} b_{j+i}2^{i-1} - \sum_{i=2}^{m+1} b_{j+i}2^{i-1}\right) \bmod p \\
&= (b_{j+1} - b_{j+m+1}2^{m}) \bmod p
\end{aligned}
$$

So we can isolate either $F_p(B_j)$ or $F_p(B_{j+1})$ and get

$$F_p(B_j) = (b_{j+1} + 2F_p(B_{j+1}) - b_{j+m+1}2^m) \bmod p$$

$$F_p(B_{j+1}) = 2^{-1}(b_{j+m+1}2^m + F_p(B_j) - b_{j+1}) \bmod p$$

Thus, by precomputing $2^m \bmod p$ and $2^{-1} \bmod p$ we can start by computing either B_{n-m} or B_0 in $\mathcal{O}(m)$ time, and then compute each of the remaining $n-m$ fingerprints in $\mathcal{O}(1)$ time each. The total time is therefore $\mathcal{O}(m+n)$.

Pattern Matching: Las Vegas I

If the Monte Carlo algorithm claims that $\mathbf{a} = \mathbf{b}[j+1 \ldots j+m]$, verify this in $\mathcal{O}(m)$ time. If false, use naive $\mathcal{O}(mn)$ time algorithm.

$$\begin{aligned}
\mathbb{E}[\text{time}] &\in \mathcal{O}(m+n) + \Pr[\text{false positive}] \cdot \mathcal{O}(mn) \\
&\subseteq \mathcal{O}(m+n) + \mathcal{O}(\tfrac{1}{n})\mathcal{O}(mn) \\
&= \mathcal{O}(m+n)
\end{aligned}$$

Good expectation, and good worst case, but bad variance. In particular, we have

$$\Pr[\text{time} = \Omega(mn)] \in \Omega(\tfrac{1}{n})$$

Pattern Matching: Las Vegas I

If the Monte Carlo algorithm claims that $\mathbf{a} = \mathbf{b}[j+1 \ldots j+m]$, verify this in $\mathcal{O}(m)$ time. If false, use naive $\mathcal{O}(mn)$ time algorithm.

$$\begin{aligned}
\mathbb{E}[\text{time}] &\in \mathcal{O}(m+n) + \Pr[\text{false positive}] \cdot \mathcal{O}(mn) \\
&\subseteq \mathcal{O}(m+n) + \mathcal{O}(\tfrac{1}{n})\mathcal{O}(mn) \\
&= \mathcal{O}(m+n)
\end{aligned}$$

Good expectation, and good worst case, but bad variance. In particular, we have

$$\Pr[\text{time} = \Omega(mn)] \in \Omega(\tfrac{1}{n})$$

Pattern Matching: Las Vegas I

If the Monte Carlo algorithm claims that $\mathbf{a} = \mathbf{b}[j+1 \dots j+m]$, verify this in $\mathcal{O}(m)$ time. If false, use naive $\mathcal{O}(mn)$ time algorithm.

$$\begin{aligned}\mathbb{E}[\text{time}] &\in \mathcal{O}(m+n) + \Pr[\text{false positive}] \cdot \mathcal{O}(mn) \\ &\subseteq \mathcal{O}(m+n) + \mathcal{O}(\tfrac{1}{n})\mathcal{O}(mn) \\ &= \mathcal{O}(m+n)\end{aligned}$$

Good expectation, and good worst case, but bad variance. In particular, we have

$$\Pr[\text{time} = \Omega(mn)] \in \Omega(\tfrac{1}{n})$$

Let $p = \Pr[\text{false positive}] \in \mathcal{O}(\frac{1}{n})$, and let $T_1 \in \mathcal{O}(m+n)$ be the time used by the Monte Carlo algorithm and $T_2 \in \mathcal{O}(mn)$ be the time used by the naive algorithm. Then the actual formula is

$$\begin{aligned}\mathbb{E}[\text{time}] &= (1-p)T_1 + p(T_1 + T_2) \\ &= T_1 + pT_2 \\ &\in \mathcal{O}(m+n) + \mathcal{O}(\tfrac{1}{n})\mathcal{O}(mn) \\ &= \mathcal{O}(m+n)\end{aligned}$$

Pattern Matching: Las Vegas I

If the Monte Carlo algorithm claims that $\mathbf{a} = \mathbf{b}[j+1 \ldots j+m]$, verify this in $\mathcal{O}(m)$ time. If false, use naive $\mathcal{O}(mn)$ time algorithm.

$$\begin{aligned}
\mathbb{E}[\text{time}] &\in \mathcal{O}(m+n) + \Pr[\text{false positive}] \cdot \mathcal{O}(mn) \\
&\subseteq \mathcal{O}(m+n) + \mathcal{O}(\tfrac{1}{n})\mathcal{O}(mn) \\
&= \mathcal{O}(m+n)
\end{aligned}$$

Good expectation, and good worst case, but bad variance. In particular, we have

$$\Pr[\text{time} = \Omega(mn)] \in \Omega(\tfrac{1}{n})$$

Pattern Matching: Las Vegas I

If the Monte Carlo algorithm claims that $\mathbf{a} = \mathbf{b}[j+1 \ldots j+m]$, verify this in $\mathcal{O}(m)$ time. If false, use naive $\mathcal{O}(mn)$ time algorithm.

$$\begin{aligned}
\mathbb{E}[\text{time}] &\in \mathcal{O}(m+n) + \Pr[\text{false positive}] \cdot \mathcal{O}(mn) \\
&\subseteq \mathcal{O}(m+n) + \mathcal{O}(\tfrac{1}{n})\mathcal{O}(mn) \\
&= \mathcal{O}(m+n)
\end{aligned}$$

Good expectation, and good worst case, but bad variance. In particular, we have

$$\Pr[\text{time} = \Omega(mn)] \in \Omega(\tfrac{1}{n})$$

Pattern Matching: Las Vegas I

If the Monte Carlo algorithm claims that $\mathbf{a} = \mathbf{b}[j+1 \ldots j+m]$, verify this in $\mathcal{O}(m)$ time. If false, use naive $\mathcal{O}(mn)$ time algorithm.

$$\begin{aligned}
\mathbb{E}[\text{time}] &\in \mathcal{O}(m+n) + \Pr[\text{false positive}] \cdot \mathcal{O}(mn) \\
&\subseteq \mathcal{O}(m+n) + \mathcal{O}(\tfrac{1}{n})\mathcal{O}(mn) \\
&= \mathcal{O}(m+n)
\end{aligned}$$

Good expectation, and good worst case, but bad variance. In particular, we have

$$\Pr[\text{time} = \Omega(mn)] \in \Omega(\tfrac{1}{n})$$

Pattern Matching: Las Vegas I

If the Monte Carlo algorithm claims that $\mathbf{a} = \mathbf{b}[j+1 \ldots j+m]$, verify this in $\mathcal{O}(m)$ time. If false, use naive $\mathcal{O}(mn)$ time algorithm.

$$\begin{aligned}
\mathbb{E}[\text{time}] &\in \mathcal{O}(m+n) + \Pr[\text{false positive}] \cdot \mathcal{O}(mn) \\
&\subseteq \mathcal{O}(m+n) + \mathcal{O}(\tfrac{1}{n})\mathcal{O}(mn) \\
&= \mathcal{O}(m+n)
\end{aligned}$$

Good expectation, and good worst case, but bad variance. In particular, we have

$$\Pr[\text{time} = \Omega(mn)] \in \Omega(\tfrac{1}{n})$$

Pattern Matching: Las Vegas II

If the Monte Carlo algorithm claims that $\mathbf{a} = \mathbf{b}[j+1 \ldots j+m]$, verify this in $\mathcal{O}(m)$ time. If false, start over with a new random p.

$$\begin{aligned} \mathbb{E}[\text{time}] &\in \frac{1}{1-\Pr[\text{false positive}]}\mathcal{O}(m+n) \\ &= \frac{1}{1-\mathcal{O}(\frac{1}{n})} \cdot \mathcal{O}(m+n) \\ &= \mathcal{O}(m+n) \end{aligned}$$

Equally good expectation, bad worst case, but good variance. In particular,

$$\Pr[\text{time} \geq k\,\mathbb{E}[\text{time}]] \in \mathcal{O}(\tfrac{1}{n^k})$$

Pattern Matching: Las Vegas II

If the Monte Carlo algorithm claims that $\mathbf{a} = \mathbf{b}[j+1 \ldots j+m]$, verify this in $\mathcal{O}(m)$ time. If false, start over with a new random p.

$$\begin{aligned}
\mathbb{E}[\text{time}] &\in \frac{1}{1-\Pr[\text{false positive}]}\mathcal{O}(m+n) \\
&= \frac{1}{1-\mathcal{O}(\frac{1}{n})} \cdot \mathcal{O}(m+n) \\
&= \mathcal{O}(m+n)
\end{aligned}$$

Equally good expectation, bad worst case, but good variance. In particular,

$$\Pr[\text{time} \geq k\,\mathbb{E}[\text{time}]] \in \mathcal{O}(\tfrac{1}{n^k})$$

Pattern Matching: Las Vegas II

If the Monte Carlo algorithm claims that $\mathbf{a} = \mathbf{b}[j+1 \ldots j+m]$, verify this in $\mathcal{O}(m)$ time. If false, start over with a new random p.

$$\begin{aligned}
\mathbb{E}[\text{time}] &\in \tfrac{1}{1-\Pr[\text{false positive}]}\mathcal{O}(m+n) \\
&= \tfrac{1}{1-\mathcal{O}(\frac{1}{n})} \cdot \mathcal{O}(m+n) \\
&= \mathcal{O}(m+n)
\end{aligned}$$

Equally good expectation, bad worst case, but good variance. In particular,

$$\Pr[\text{time} \geq k\,\mathbb{E}[\text{time}]] \in \mathcal{O}(\tfrac{1}{n^k})$$

Pattern Matching: Las Vegas II

If the Monte Carlo algorithm claims that $\mathbf{a} = \mathbf{b}[j+1 \ldots j+m]$, verify this in $\mathcal{O}(m)$ time. If false, start over with a new random p.

$$\begin{aligned}\mathbb{E}[\text{time}] &\in \frac{1}{1-\Pr[\text{false positive}]}\mathcal{O}(m+n)\\ &= \frac{1}{1-\mathcal{O}(\frac{1}{n})} \cdot \mathcal{O}(m+n)\\ &= \mathcal{O}(m+n)\end{aligned}$$

Equally good expectation, bad worst case, but good variance. In particular,

$$\Pr[\text{time} \geq k\,\mathbb{E}[\text{time}]] \in \mathcal{O}(\tfrac{1}{n^k})$$

Pattern Matching: Las Vegas II

If the Monte Carlo algorithm claims that $\mathbf{a} = \mathbf{b}[j+1 \ldots j+m]$, verify this in $\mathcal{O}(m)$ time. If false, start over with a new random p.

$$\begin{aligned}
\mathbb{E}[\text{time}] &\in \frac{1}{1-\Pr[\text{false positive}]}\mathcal{O}(m+n) \\
&= \frac{1}{1-\mathcal{O}(\frac{1}{n})} \cdot \mathcal{O}(m+n) \\
&= \mathcal{O}(m+n)
\end{aligned}$$

Equally good expectation, bad worst case, but good variance. In particular,

$$\Pr[\text{time} \geq k\,\mathbb{E}[\text{time}]] \in \mathcal{O}(\tfrac{1}{n^k})$$

Pattern Matching: Las Vegas II

If the Monte Carlo algorithm claims that $\mathbf{a} = \mathbf{b}[j+1 \ldots j+m]$, verify this in $\mathcal{O}(m)$ time. If false, start over with a new random p.

$$\begin{aligned}\mathbb{E}[\text{time}] &\in \frac{1}{1-\Pr[\text{false positive}]}\mathcal{O}(m+n)\\ &= \frac{1}{1-\mathcal{O}(\frac{1}{n})} \cdot \mathcal{O}(m+n)\\ &= \mathcal{O}(m+n)\end{aligned}$$

Equally good expectation, bad worst case, but good variance. In particular,

$$\Pr[\text{time} \geq k\,\mathbb{E}[\text{time}]] \in \mathcal{O}(\tfrac{1}{n^k})$$

Pattern Matching: Las Vegas II

If the Monte Carlo algorithm claims that $\mathbf{a} = \mathbf{b}[j+1 \ldots j+m]$, verify this in $\mathcal{O}(m)$ time. If false, start over with a new random p.

$$\begin{aligned}\mathbb{E}[\text{time}] &\in \tfrac{1}{1-\Pr[\text{false positive}]}\mathcal{O}(m+n)\\ &= \tfrac{1}{1-\mathcal{O}(\frac{1}{n})} \cdot \mathcal{O}(m+n)\\ &= \mathcal{O}(m+n)\end{aligned}$$

Equally good expectation, bad worst case, but good variance. In particular,

$$\Pr[\text{time} \geq k\,\mathbb{E}[\text{time}]] \in \mathcal{O}(\tfrac{1}{n^k})$$

Summary

- We have seen a number of *fingerprinting* techniques, mostly based on algebra.
- We have seen how these can be used to do probabilistic verification of e.g. Matrix Multiplication and Polynomial Identities.
- We saw an example of how an abstract polynomial equation can solve a concrete problem (is there a perfect matching).
- And we saw how fingerprinting can be used for string comparison, and how a special version can be used for simple Monte Carlo string search.
- Finally, we saw two different ways a monte carlo algorithm can be turned into a Las Vegas algorithm.
- Next time: Streaming Algorithms.

Summary

- We have seen a number of *fingerprinting* techniques, mostly based on algebra.
- We have seen how these can be used to do probabilistic verification of e.g. Matrix Multiplication and Polynomial Identities.
- We saw an example of how an abstract polynomial equation can solve a concrete problem (is there a perfect matching).
- And we saw how fingerprinting can be used for string comparison, and how a special version can be used for simple Monte Carlo string search.
- Finally, we saw two different ways a monte carlo algorithm can be turned into a Las Vegas algorithm.
- Next time: Streaming Algorithms.

Summary

- We have seen a number of *fingerprinting* techniques, mostly based on algebra.
- We have seen how these can be used to do probabilistic verification of e.g. Matrix Multiplication and Polynomial Identities.
- We saw an example of how an abstract polynomial equation can solve a concrete problem (is there a perfect matching).
- And we saw how fingerprinting can be used for string comparison, and how a special version can be used for simple Monte Carlo string search.
- Finally, we saw two different ways a monte carlo algorithm can be turned into a Las Vegas algorithm.
- Next time: Streaming Algorithms.

Summary

- We have seen a number of *fingerprinting* techniques, mostly based on algebra.
- We have seen how these can be used to do probabilistic verification of e.g. Matrix Multiplication and Polynomial Identities.
- We saw an example of how an abstract polynomial equation can solve a concrete problem (is there a perfect matching).
- And we saw how fingerprinting can be used for string comparison, and how a special version can be used for simple Monte Carlo string search.
- Finally, we saw two different ways a monte carlo algorithm can be turned into a Las Vegas algorithm.
- Next time: Streaming Algorithms.

Summary

- We have seen a number of *fingerprinting* techniques, mostly based on algebra.
- We have seen how these can be used to do probabilistic verification of e.g. Matrix Multiplication and Polynomial Identities.
- We saw an example of how an abstract polynomial equation can solve a concrete problem (is there a perfect matching).
- And we saw how fingerprinting can be used for string comparison, and how a special version can be used for simple Monte Carlo string search.
- Finally, we saw two different ways a monte carlo algorithm can be turned into a Las Vegas algorithm.
- Next time: Streaming Algorithms.

Summary

- We have seen a number of *fingerprinting* techniques, mostly based on algebra.
- We have seen how these can be used to do probabilistic verification of e.g. Matrix Multiplication and Polynomial Identities.
- We saw an example of how an abstract polynomial equation can solve a concrete problem (is there a perfect matching).
- And we saw how fingerprinting can be used for string comparison, and how a special version can be used for simple Monte Carlo string search.
- Finally, we saw two different ways a monte carlo algorithm can be turned into a Las Vegas algorithm.
- Next time: Streaming Algorithms.

Summary

- We have seen a number of *fingerprinting* techniques, mostly based on algebra.
- We have seen how these can be used to do probabilistic verification of e.g. Matrix Multiplication and Polynomial Identities.
- We saw an example of how an abstract polynomial equation can solve a concrete problem (is there a perfect matching).
- And we saw how fingerprinting can be used for string comparison, and how a special version can be used for simple Monte Carlo string search.
- Finally, we saw two different ways a monte carlo algorithm can be turned into a Las Vegas algorithm.
- Next time: Streaming Algorithms.