Nisan's Pseudorandom Generator for RL

$\mathsf{BPL} \subseteq \mathsf{SC} = \mathsf{DTISP}(\mathrm{poly}(n), \log^2(n))$

Soham Chatterjee

December 2, 2025

Pesudorandomness Course (CSS.413.1) Presentation, STCS

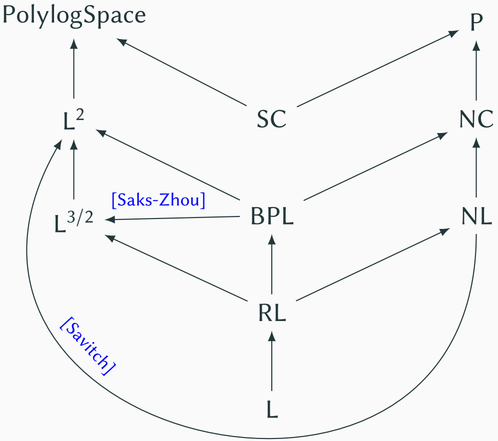
Complexity Classes

- L: Deterministic Logarithmic Space.
- L^α, $\alpha > 0$: Set of problems decidable in $O(\log^\alpha n)$ space deterministically.
- NL: Nondeterministic Logarithmic Space.
- RL: Randomized Logarithmic Space with One-sided error $\frac{1}{3}$.
- BPL: Randomized Logarithmic Space with Two-sided error $\frac{1}{3}$.
- SC: Steve's Class or $\mathsf{DTISP}(\mathrm{poly}(n), \mathrm{poly}(\log n))$ i.e. set of problems decidable deterministically in polynomial time and polylog space.
- NC: Nick's Class i.e. set of problems decidable in circuits of polynomial size and polylog depth and bounded fan-in.

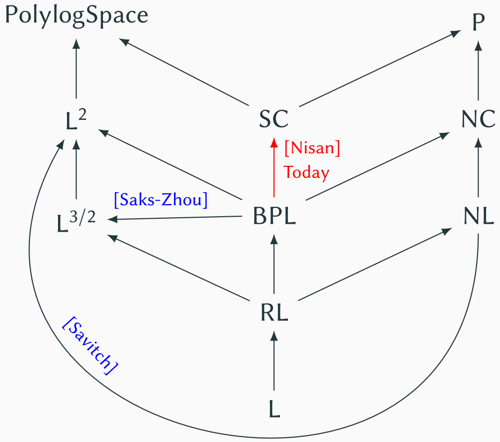
Remark

Don't confuse SC with $\mathsf{P} \cap \mathsf{PolylogSpace}$!

Complexity Classes Zoo



Complexity Classes Zoo



Pseudorandom Generator

Definition (Pseudorandom Generator)

A map $\mathcal{G} : \{0,1\}^l \to \{0,1\}^n$, where $n \geq l$ is called a PRG for a class $\mathcal{C}$ with a parameter $\epsilon > 0$ if for any $f \in \mathcal{C}$,

$$\left| \underset{y \in \{0,1\}^n}{\mathbb{P}}[f(y) = 1] - \underset{x \in \{0,1\}^l}{\mathbb{P}}[f(\mathcal{G}(x)) = 1] \right| \leq \epsilon$$

- Here l is called the seed-length of the PRG.
- $n - l$ is called the stretch of the PRG.

Pseudorandom Generator

Definition (Pseudorandom Generator)

A map $\mathcal{G} : \{0,1\}^l \to \{0,1\}^n$, where $n \geq l$ is called a PRG for a class $\mathcal{C}$ with a parameter $\epsilon > 0$ if for any $f \in \mathcal{C}$,

$$\left| \mathbb{P}_{y \in \{0,1\}^n}[f(y) = 1] - \mathbb{P}_{x \in \{0,1\}^l}[f(\mathcal{G}(x)) = 1] \right| \leq \epsilon$$

- Here l is called the seed-length of the PRG.
- $n - l$ is called the stretch of the PRG.
- We call $\mathcal{G}$, ϵ-fools $\mathcal{C}$.

Pseudorandom Generator

Definition (Pseudorandom Generator)

A map $\mathcal{G} : \{0,1\}^l \to \{0,1\}^n$, where $n \geq l$ is called a PRG for a class $\mathcal{C}$ with a parameter $\epsilon > 0$ if for any $f \in \mathcal{C}$,

$$\left| \underset{y \in \{0,1\}^n}{\mathbb{P}}[f(y) = 1] - \underset{x \in \{0,1\}^l}{\mathbb{P}}[f(\mathcal{G}(x)) = 1] \right| \leq \epsilon$$

- Here l is called the seed-length of the PRG.
- $n - l$ is called the stretch of the PRG.
- We call $\mathcal{G}$, ϵ-fools $\mathcal{C}$.
- Typically, we want $n >> l$ and $\mathcal{G}$ to be efficiently computable.

Finite State Automata

Let T be a BPL machine which uses n^c random bits on inputs of length n and runs in polynomial time and uses $S = O(\log n)$ space.

Finite State Automata

Let T be a BPL machine which uses n^c random bits on inputs of length n and runs in polynomial time and uses $S = O(\log n)$ space.

- There are at most $N := 2^{O(S)} = \mathsf{poly}(n)$ configurations of T.

Finite State Automata

Let T be a BPL machine which uses n^c random bits on inputs of length n and runs in polynomial time and uses $S = O(\log n)$ space.

- There are at most $N := 2^{O(S)} = \mathsf{poly}(n)$ configurations of T.
- Each random bit is used to make a transition between two configurations.

Finite State Automata

Let T be a BPL machine which uses n^c random bits on inputs of length n and runs in polynomial time and uses $S = O(\log n)$ space.

- There are at most $N := 2^{O(S)} = \mathrm{poly}(n)$ configurations of T.
- Each random bit is used to make a transition between two configurations.
- The starting configuration is fixed for any input.

Finite State Automata

Let T be a BPL machine which uses n^c random bits on inputs of length n and runs in polynomial time and uses $S = O(\log n)$ space.

- There are at most $N := 2^{O(S)} = \mathrm{poly}(n)$ configurations of T.
- Each random bit is used to make a transition between two configurations.
- The starting configuration is fixed for any input.
- Input x is accepted if T reaches a state representing acceptance.

Finite State Automata

Let T be a BPL machine which uses n^c random bits on inputs of length n and runs in polynomial time and uses $S = O(\log n)$ space.

- There are at most $N := 2^{O(S)} = \text{poly}(n)$ configurations of T.
- Each random bit is used to make a transition between two configurations.
- The starting configuration is fixed for any input.
- Input x is accepted if T reaches a state representing acceptance.

Therefore the configuration graph of T on input x represents a finite state automata with N states.

Computational Tableau of BPL machine

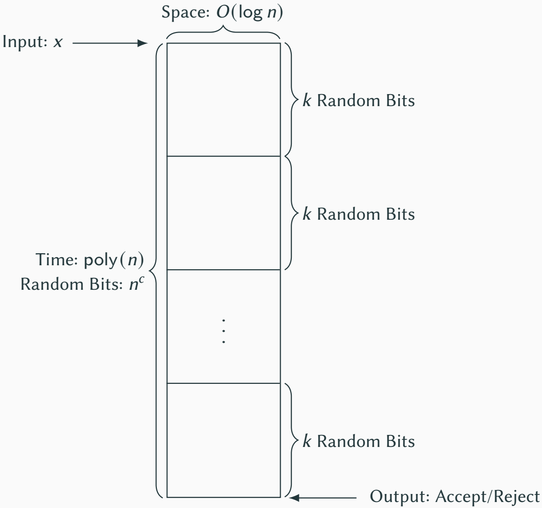


Figure 1: Computational Tableau of BPL machine T.

Dividing BPL Computation into Blocks

- Let the BPL machine T run in time $\mathsf{poly}(n)$ using n^c random bits and $O(\log n)$ space on input x of length n.

Dividing BPL Computation into Blocks

- Let the BPL machine T run in time $\mathsf{poly}(n)$ using n^c random bits and $O(\log n)$ space on input x of length n.
- Let $k << n^c$ be a parameter to be fixed later.
- Divide the computation of T into $t = n^c/k$ blocks, where each block uses k random bits.

Dividing BPL Computation into Blocks

- Let the BPL machine T run in time $\mathsf{poly}(n)$ using n^c random bits and $O(\log n)$ space on input x of length n.
- Let $k << n^c$ be a parameter to be fixed later.
- Divide the computation of T into $t = n^c/k$ blocks, where each block uses k random bits.
- We can treat each block as a separate BPL machine T_i (in some sense) which takes input as the final configuration of T_{i-1} and k random bits.

FSA for Computation Blocks

- We can think of T as a finite state automata with N states.
- Each state makes 2^k transitions where each transion corresponds to a choice of k random bits.

FSA for Computation Blocks

- We can think of T as a finite state automata with N states.
- Each state makes 2^k transitions where each transion corresponds to a choice of k random bits.
- Let Q be this automata. For transition we write $Q(i; r) = j$ for any $r \in \{0, 1\}^k$ if Q goes from state i to state j when fed r.

FSA for Computation Blocks

- We can think of T as a finite state automata with N states.
- Each state makes 2^k transitions where each transion corresponds to a choice of k random bits.
- Let Q be this automata. For transition we write $Q(i; r) = j$ for any $r \in \{0, 1\}^k$ if Q goes from state i to state j when fed r.
- Let M be the matrix where $M[i, j] = \underset{r \in \{0,1\}^k}{\mathbb{P}} [Q(i; x) = j]$

FSA for Computation Blocks

- We can think of T as a finite state automata with N states.
- Each state makes 2^k transitions where each transion corresponds to a choice of k random bits.
- Let Q be this automata. For transition we write $Q(i; r) = j$ for any $r \in \{0, 1\}^k$ if Q goes from state i to state j when fed r.
- Let M be the matrix where $M[i, j] = \mathbb{P}_{r \in \{0,1\}^k}[Q(i; x) = j]$

$$\mathbb{P}_{r \in \{0,1\}^{k \times t}}[T(x, r) = \mathrm{Acc}] = \sum_{j:\text{Accepting state}} M^t[1, j]$$

Goal: Approximate M^t using PRG.

Approximate Automata Matrix

Suppose we have a pseudorandom generator $\mathcal{G} : \{0,1\}^k \to \{0,1\}^{k \cdot t}$. Let Q be a finite state automata with N states and its matrix be M as defined above.

Approximate Automata Matrix

Suppose we have a pseudorandom generator $\mathcal{G} : \{0,1\}^k \to \{0,1\}^{k \cdot t}$. Let Q be a finite state automata with N states and its matrix be M as defined above.

- From definition of M, $M^t[i,j] = \underset{r_1,\ldots,r_t \in \{0,1\}^k}{\mathbb{P}}[Q(i; r_1 \ldots; r_t) = j]$

Approximate Automata Matrix

Suppose we have a pseudorandom generator $\mathcal{G} : \{0,1\}^k \to \{0,1\}^{k \cdot t}$. Let Q be a finite state automata with N states and its matrix be M as defined above.

- From definition of M, $M^t[i,j] = \underset{r_1,\ldots,r_t \in \{0,1\}^k}{\mathbb{P}}[Q(i; r_1 \ldots; r_t) = j]$
- Using $\mathcal{G}$, let $M_{\mathcal{G}}[i,j] = \underset{r \in \{0,1\}^k}{\mathbb{P}}[Q(i; \mathcal{G}(r)) = j]$

Approximate Automata Matrix

Suppose we have a pseudorandom generator $\mathcal{G} : \{0,1\}^k \to \{0,1\}^{k \cdot t}$. Let Q be a finite state automata with N states and its matrix be M as defined above.

- From definition of M, $M^t[i,j] = \underset{r_1,\ldots,r_t \in \{0,1\}^k}{\mathbb{P}}[Q(i; r_1 \ldots; r_t) = j]$
- Using $\mathcal{G}$, let $M_{\mathcal{G}}[i,j] = \underset{r \in \{0,1\}^k}{\mathbb{P}}[Q(i; \mathcal{G}(r)) = j]$
- Want to construct $\mathcal{G}$ such that

$$\|M^t - M_{\mathcal{G}}\| < \epsilon$$

for small ϵ.

Approximate Automata Matrix

Suppose we have a pseudorandom generator $\mathcal{G} : \{0,1\}^k \to \{0,1\}^{k \cdot t}$. Let Q be a finite state automata with N states and its matrix be M as defined above.

- From definition of M, $M^t[i,j] = \underset{r_1,\ldots,r_t \in \{0,1\}^k}{\mathbb{P}}[Q(i; r_1 \ldots; r_t) = j]$
- Using $\mathcal{G}$, let $M_{\mathcal{G}}[i,j] = \underset{r \in \{0,1\}^k}{\mathbb{P}}[Q(i; \mathcal{G}(r)) = j]$
- Want to construct $\mathcal{G}$ such that

  $$\|M^t - M_{\mathcal{G}}\| < \epsilon$$

  for small ϵ.

Then if T decides a language with error probability at most $\frac{1}{3}$, using $\mathcal{G}$ we can calculate the $\sum_{j:\text{Accepting state}} M_{\mathcal{G}}[1,j]$ and decide the language if it is at least $\frac{2}{3} - \epsilon$.

Matrix Norm

For any vector $v \in \mathbb{R}^N$, define $\|v\| = \sum_{i \in [N]} |v(i)|$. Then for any matrix $M \in \mathbb{R}^{N \times N}$ define $\|M\| = \sup_{0 \neq v \in \mathbb{R}^N} \frac{\|Mv\|}{\|v\|}$

Properties:

- $\|M\| \leq \max_{i \in [N]} \sum_{j \in [N]} |M[i, j]|$
- $\|M + N\| \leq \|M\| + \|N\|$
- $\|MN\| \leq \|M\| \cdot \|N\|$

Universal Hash Family

Definition (Universal Hash Family (Carter-Wegman))

$\mathcal{H} = \{h : \{0,1\}^k \to \{0,1\}^m\}$ is a *universal hash family* if for any $x_1 \neq x_2 \in \{0,1\}^k$ and $y_1, y_2 \in \{0,1\}^m$,

- $\underset{h \in \mathcal{H}}{\mathbb{P}}[\, h(x_1) = y_1 \,] \;=\; \frac{1}{2^m}$
- $\underset{h \in \mathcal{H}}{\mathbb{P}}[\, h(x_1) = y_1 \;\wedge\; h(x_2) = y_2 \,] \;=\; \frac{1}{2^{2m}}$

Universal Hash Family

Definition (Universal Hash Family (Carter-Wegman))

$\mathcal{H} = \{h : \{0,1\}^k \longrightarrow \{0,1\}^m\}$ is a *universal hash family* if for any $x_1 \neq x_2 \in \{0,1\}^k$ and $y_1, y_2 \in \{0,1\}^m$,

- $\underset{h \in \mathcal{H}}{\mathbb{P}}[\, h(x_1) = y_1 \,] \;=\; \frac{1}{2^m}$
- $\underset{h \in \mathcal{H}}{\mathbb{P}}[\, h(x_1) = y_1 \;\wedge\; h(x_2) = y_2 \,] \;=\; \frac{1}{2^{2m}}$

- For our purpose, we have $k = m$.

Universal Hash Family

Definition (Universal Hash Family (Carter-Wegman))

$\mathcal{H} = \{h : \{0,1\}^k \to \{0,1\}^m\}$ is a *universal hash family* if for any $x_1 \neq x_2 \in \{0,1\}^k$ and $y_1, y_2 \in \{0,1\}^m$,

- $\underset{h \in \mathcal{H}}{\mathbb{P}}[\, h(x_1) = y_1 \,] \;=\; \frac{1}{2^m}$
- $\underset{h \in \mathcal{H}}{\mathbb{P}}[\, h(x_1) = y_1 \;\wedge\; h(x_2) = y_2 \,] \;=\; \frac{1}{2^{2m}}$

- For our purpose, we have $k = m$.
- We can construct such a family with $|\mathcal{H}| = 2^{O(k)}$ where

$$\mathcal{H} = \{a \cdot x + b \mid a, b \in \{0,1\}^k\}$$

over $GF(2^k)$.

Property of Universal Hash Family

Definition ((ϵ, A, B)-good hash function)

Let $A \subseteq \{0,1\}^k$, $B \subseteq \{0,1\}^m$, $\epsilon > 0$, $h : \{0,1\}^k \longrightarrow \{0,1\}^m$ is said to be (ϵ, A, B)-*good* if

$$\left| \mathbb{P}_{x \in \{0,1\}^k}[x \in A \wedge h(x) \in B] - \alpha \cdot \beta \right| \leq \epsilon$$

where $\alpha = \frac{|A|}{2^k}$ and $\beta = \frac{|B|}{2^m}$.

Property of Universal Hash Family

Definition ((ϵ, A, B)-good hash function)

Let $A \subseteq \{0,1\}^k$, $B \subseteq \{0,1\}^m$, $\epsilon > 0$, $h : \{0,1\}^k \to \{0,1\}^m$ is said to be *(ϵ, A, B)-good* if

$$\left| \underset{x \in \{0,1\}^k}{\mathbb{P}}[x \in A \wedge h(x) \in B] - \alpha \cdot \beta \right| \leq \epsilon$$

where $\alpha = \frac{|A|}{2^k}$ and $\beta = \frac{|B|}{2^m}$.

Lemma (Proved in Appendix)

If $\mathcal{H}$ is a universal hash family, then for any $A \subseteq \{0,1\}^k$, $B \subseteq \{0,1\}^m$, $\epsilon > 0$,

$$\underset{h \in \mathcal{H}}{\mathbb{P}}[h \text{ is not } (\epsilon, A, B)\text{-good}] \leq \frac{\alpha \cdot \beta}{2^k \epsilon^2}$$

Nisan's Generator

Let $\mathcal{H}$ be an universal hash family from $\{0,1\}^k$ to $\{0,1\}^k$. For any integer $m \geq 0$ define the function $\mathcal{G}_m \colon \{0,1\}^k \times \mathcal{H}^m \to \{0,1\}^{k \cdot 2^m}$ recursively as follows:

Nisan's Generator

Let $\mathcal{H}$ be an universal hash family from $\{0,1\}^k$ to $\{0,1\}^k$. For any integer $m \geq 0$ define the function $\mathcal{G}_m \colon \{0,1\}^k \times \mathcal{H}^m \to \{0,1\}^{k \cdot 2^m}$ recursively as follows:

- $\mathcal{G}_0(x) = x$
- $\mathcal{G}_m(x, h_1, \ldots, h_m) = (\mathcal{G}_{m-1}(x, h_1, \ldots, h_{m-1}), \mathcal{G}_{m-1}(h_m(x), h_1, \ldots, h_{m-1}))$

Nisan's Generator

Let $\mathcal{H}$ be an universal hash family from $\{0,1\}^k$ to $\{0,1\}^k$. For any integer $m \geq 0$ define the function $\mathcal{G}_m \colon \{0,1\}^k \times \mathcal{H}^m \to \{0,1\}^{k \cdot 2^m}$ recursively as follows:

- $\mathcal{G}_0(x) = x$
- $\mathcal{G}_m(x, h_1, \ldots, h_m) = (\mathcal{G}_{m-1}(x, h_1, \ldots, h_{m-1}), \mathcal{G}_{m-1}(h_m(x), h_1, \ldots, h_{m-1}))$

For example:

$$\mathcal{G}_1(x, h) = (x, h(x)), \quad \mathcal{G}_2(x, h_1, h_2) = (x, h_1(x), h_2(x), h_1 \cdot h_2(x))$$

Nisan's Generator

Let $\mathcal{H}$ be an universal hash family from $\{0,1\}^k$ to $\{0,1\}^k$. For any integer $m \geq 0$ define the function $\mathcal{G}_m \colon \{0,1\}^k \times \mathcal{H}^m \to \{0,1\}^{k \cdot 2^m}$ recursively as follows:

- $\mathcal{G}_0(x) = x$
- $\mathcal{G}_m(x, h_1, \ldots, h_m) = (\mathcal{G}_{m-1}(x, h_1, \ldots, h_{m-1}), \mathcal{G}_{m-1}(h_m(x), h_1, \ldots, h_{m-1}))$

For example:

$$\mathcal{G}_1(x, h) = (x, h(x)), \quad \mathcal{G}_2(x, h_1, h_2) = (x, h_1(x), h_2(x), h_1 \cdot h_2(x))$$

$$\mathcal{G}_3(x, h_1, h_2, h_3) = (x, h_1(x), h_2(x), h_1 \cdot h_2(x), h_3(x), h_1 \cdot h_3(x), h_2 \cdot h_3(x), h_1 \cdot h_2 \cdot h_3($$

Nisan's Generator

Let $\mathcal{H}$ be an universal hash family from $\{0,1\}^k$ to $\{0,1\}^k$. For any integer $m \geq 0$ define the function $\mathcal{G}_m \colon \{0,1\}^k \times \mathcal{H}^m \to \{0,1\}^{k \cdot 2^m}$ recursively as follows:

- $\mathcal{G}_0(x) = x$
- $\mathcal{G}_m(x, h_1, \ldots, h_m) = (\mathcal{G}_{m-1}(x, h_1, \ldots, h_{m-1}), \mathcal{G}_{m-1}(h_m(x), h_1, \ldots, h_{m-1}))$

For example:

$$\mathcal{G}_1(x, h) = (x, h(x)), \quad \mathcal{G}_2(x, h_1, h_2) = (x, h_1(x), h_2(x), h_1 \cdot h_2(x))$$

$$\mathcal{G}_3(x, h_1, h_2, h_3) = (x, h_1(x), h_2(x), h_1 \cdot h_2(x), h_3(x), h_1 \cdot h_3(x), h_2 \cdot h_3(x), h_1 \cdot h_2 \cdot h_3($$

- We want $k \cdot 2^m = n^c \implies m = \log t$.
- This gives a stretch of $k \cdot (t-1)$ bits.

Proof Flow

Let $h_1, \ldots, h_s$ be some fixed hash functions from $\mathcal{H}$. Define the matrix

$$M_{h_1,\ldots,h_s}[i,j] = \mathop{\mathbb{P}}_{x \in \{0,1\}^k}[Q(i; \mathcal{G}_s(x, h_1, \ldots, h_s)) = j]$$

- Using $h_1, \ldots, h_s$ we had 2^s many transitions in Q. So we should compare $M_{h_1,\ldots,h_s}$ with M^{2^s}.

Proof Flow

Let $h_1, \ldots, h_s$ be some fixed hash functions from $\mathcal{H}$. Define the matrix

$$M_{h_1,\ldots,h_s}[i,j] = \mathop{\mathbb{P}}_{x \in \{0,1\}^k}[Q(i; \mathcal{G}_s(x, h_1, \ldots, h_s)) = j]$$

- Using $h_1, \ldots, h_s$ we had 2^s many transitions in Q. So we should compare $M_{h_1,\ldots,h_s}$ with M^{2^s}.

Goal: For 'good' choice of $h_1, \ldots, h_m$, $\|M^{2^m} - M_{h_1,\ldots,h_m}\| < \epsilon$

Proof Flow

Let $h_1, \ldots, h_s$ be some fixed hash functions from $\mathcal{H}$. Define the matrix

$$M_{h_1,\ldots,h_s}[i,j] = \mathbb{P}_{x \in \{0,1\}^k}[Q(i; \mathcal{G}_s(x, h_1, \ldots, h_s)) = j]$$

- Using $h_1, \ldots, h_s$ we had 2^s many transitions in Q. So we should compare $M_{h_1,\ldots,h_s}$ with M^{2^s}.

Goal: For 'good' choice of $h_1, \ldots, h_m$, $\|M^{2^m} - M_{h_1,\ldots,h_m}\| < \epsilon$

Approach:

Step 1: Suppose we have $h_1, \ldots, h_{s-1}$. We will find $h_s \in \mathcal{H}$ such that for all $i, j \in [N]$,

$$\left\| M^2_{h_1,\ldots,h_{s-1}} - M_{h_1,\ldots,h_s} \right\| \leq \delta$$

Proof Flow

Let $h_1, \ldots, h_s$ be some fixed hash functions from $\mathcal{H}$. Define the matrix

$$M_{h_1,\ldots,h_s}[i,j] = \underset{x \in \{0,1\}^k}{\mathbb{P}}[Q(i; \mathcal{G}_s(x, h_1, \ldots, h_s)) = j]$$

- Using $h_1, \ldots, h_s$ we had 2^s many transitions in Q. So we should compare $M_{h_1,\ldots,h_s}$ with M^{2^s}.

Goal: For 'good' choice of $h_1, \ldots, h_m$, $\|M^{2^m} - M_{h_1,\ldots,h_m}\| < \epsilon$

Approach:

Step 1: Suppose we have $h_1, \ldots, h_{s-1}$. We will find $h_s \in \mathcal{H}$ such that for all $i, j \in [N]$,

$$\left\| M^2_{h_1,\ldots,h_{s-1}} - M_{h_1,\ldots,h_s} \right\| \leq \delta$$

Step 2: Using above property will show for all $s \in [m]$,

$$\left\| M_{h_1,\ldots,h_s} - M^{2^s} \right\| \leq (2^s - 1)\delta$$

Find good h_s from $h_1, \ldots, h_{s-1}$

Suppose we have $h_1, \ldots, h_{s-1} \in \mathcal{H}$ such that,

$$\left\| M_{h_1,\ldots,h_{ls-1}} - M^{2^{s-1}} \right\| \leq (2^{s-1} - 1)\delta$$

If we can find h_s such that $\|M^2_{h_1,\ldots,h_{s-1}} - M_{h_1,\ldots,h_s}\| \leq \delta$ then we are done.

Find good h_s from $h_1, \ldots, h_{s-1}$

Suppose we have $h_1, \ldots, h_{s-1} \in \mathcal{H}$ such that,

$$\left\| \mathcal{M}_{h_1,\ldots,h_{ls-1}} - \mathcal{M}^{2^{s-1}} \right\| \leq (2^{s-1} - 1)\delta$$

If we can find h_s such that $\|\mathcal{M}^2_{h_1,\ldots,h_{s-1}} - \mathcal{M}_{h_1,\ldots,h_s}\| \leq \delta$ then we are done.

Algorithm (Find): Go over all $h \in \mathcal{H}$ and all $i, j \in [N]$:

Find good h_s from $h_1, \ldots, h_{s-1}$

Suppose we have $h_1, \ldots, h_{s-1} \in \mathcal{H}$ such that,

$$\left\| M_{h_1,\ldots,h_{ls-1}} - M^{2^{s-1}} \right\| \leq (2^{s-1} - 1)\delta$$

If we can find h_s such that $\|M^2_{h_1,\ldots,h_{s-1}} - M_{h_1,\ldots,h_s}\| \leq \delta$ then we are done.

Algorithm (Find): Go over all $h \in \mathcal{H}$ and all $i, j \in [N]$:

Step 1: Compute

- $p_1 = M_{h_1,\ldots,h_{s-1},h}[i, j]$
- $p_2 = \sum_{l \in [N]} M_{h_1,\ldots,h_{s-1}}[i, l] \cdot M_{h_1,\ldots,h_{s-1}}[l, j]$

Find good h_s from $h_1, \ldots, h_{s-1}$

Suppose we have $h_1, \ldots, h_{s-1} \in \mathcal{H}$ such that,

$$\left\| M_{h_1,\ldots,h_{ls-1}} - M^{2^{s-1}} \right\| \leq (2^{s-1} - 1)\delta$$

If we can find h_s such that $\|M^2_{h_1,\ldots,h_{s-1}} - M_{h_1,\ldots,h_s}\| \leq \delta$ then we are done.

Algorithm (Find): Go over all $h \in \mathcal{H}$ and all $i, j \in [N]$:

Step 1: Compute

- $p_1 = M_{h_1,\ldots,h_{s-1},h}[i, j]$
- $p_2 = \sum_{l \in [N]} M_{h_1,\ldots,h_{s-1}}[i, l] \cdot M_{h_1,\ldots,h_{s-1}}[l, j]$

Step 2: Check if $|p_1 - p_2| > \frac{\delta}{N}$ go to next h else return h.

Find good h_s from $h_1, \ldots, h_{s-1}$

Suppose we have $h_1, \ldots, h_{s-1} \in \mathcal{H}$ such that,

$$\left\| M_{h_1,\ldots,h_{l_{s-1}}} - M^{2^{s-1}} \right\| \leq (2^{s-1} - 1)\delta$$

If we can find h_s such that $\|M^2_{h_1,\ldots,h_{s-1}} - M_{h_1,\ldots,h_s}\| \leq \delta$ then we are done.

Algorithm (Find): Go over all $h \in \mathcal{H}$ and all $i, j \in [N]$:

Step 1: Compute

- $p_1 = M_{h_1,\ldots,h_{s-1},h}[i,j]$
- $p_2 = \sum_{l \in [N]} M_{h_1,\ldots,h_{s-1}}[i,l] \cdot M_{h_1,\ldots,h_{s-1}}[l,j]$

Step 2: Check if $|p_1 - p_2| > \frac{\delta}{N}$ go to next h else return h.

Remark

To compute $M_{h_1,\ldots,h_{s-1},h}[i,j]$ it goes over all $r \in \{0,1\}^k$ and compute $\mathcal{G}_s(r; h)1, \ldots, h_s)$ and counts how many r gives $Q(i; \mathcal{G}_s(r, h_1, \ldots, h_s)) = j$. and return $count/2^k$.

Algorithm always returns an h (I)

Claim

There exists an $h_s \in \mathcal{H}$ *such that for all* $i, j \in [N]$,

$$\left| M_{h_1,\ldots,h_{s-1},h_s}[i,j] - M^2_{h_1,\ldots,h_{s-1}}[i,j] \right| \leq \frac{\delta}{N}$$

Algorithm always returns an h (I)

Claim

There exists an $h_s \in \mathcal{H}$ such that for all $i, j \in [N]$,

$$\left| M_{h_1,\ldots,h_{s-1},h_s}[i,j] - M^2_{h_1,\ldots,h_{s-1}}[i,j] \right| \leq \frac{\delta}{N}$$

Let $A_{i,j}$ be the set of $r \in \{0,1\}^k$ such that $Q(i; \mathcal{G}_{s-1}(r, h_1, \ldots, h_{s-1})) = j$. So $M_{h_1,\ldots,h_{s-1}}[i,j] = \rho(A_{i,j})$ where $\rho(A) = |A|/2^k$.

Algorithm always returns an h (I)

Claim

There exists an $h_s \in \mathcal{H}$ such that for all $i, j \in [N]$,

$$\left| M_{h_1,\ldots,h_{s-1},h_s}[i,j] - M^2_{h_1,\ldots,h_{s-1}}[i,j] \right| \leq \frac{\delta}{N}$$

Let $A_{i,j}$ be the set of $r \in \{0,1\}^k$ such that $Q(i; \mathcal{G}_{s-1}(r, h_1, \ldots, h_{s-1})) = j$. So $M_{h_1,\ldots,h_{s-1}}[i,j] = \rho(A_{i,j})$ where $\rho(A) = |A|/2^k$.

- For any $i, j \in [N]$,

$$M^2_{h_1,\ldots,h_{s-1}} = \sum_{l \in [N]} \rho(A_{i,l}) \cdot \rho(A_{l,j})$$

Algorithm always returns an h (I)

Claim

There exists an $h_s \in \mathcal{H}$ such that for all $i, j \in [N]$,

$$\left| M_{h_1,\ldots,h_{s-1},h_s}[i,j] - M^2_{h_1,\ldots,h_{s-1}}[i,j] \right| \leq \frac{\delta}{N}$$

Let $A_{i,j}$ be the set of $r \in \{0,1\}^k$ such that $Q(i; \mathcal{G}_{s-1}(r, h_1, \ldots, h_{s-1})) = j$. So $M_{h_1,\ldots,h_{s-1}}[i,j] = \rho(A_{i,j})$ where $\rho(A) = |A|/2^k$.

- For any $i, j \in [N]$,

$$M^2_{h_1,\ldots,h_{s-1}} = \sum_{l \in [N]} \rho(A_{i,l}) \cdot \rho(A_{l,j})$$

- For any $h \in \mathcal{H}$,

$$M_{h_1,\ldots,h_{s-1},h}[i,j] = \sum_{l \in [N]} \mathbb{P}_{r \in \{0,1\}^k}[r \in A_{i,l} \wedge h(r) \in A_{l,j}]$$

Algorithm always returns an h (II)

For a random $h \in \mathcal{H}$ with probability at least $1 - \frac{N^4}{2^k \delta^2} \geq 1 - \frac{1}{2n^3}$,

$$\left| \underset{r \in \{0,1\}^k}{\mathbb{P}} [r \in A_{i,l} \wedge h(r) \in A_{l,j}] - \rho(A_{i,l}) \cdot \rho(A_{l,j}) \right| \leq \frac{\rho(A_{i,l}) \cdot \rho(A_{l,j})}{k^2} \leq \frac{\delta}{N^2}$$

Algorithm always returns an h (II)

For a random $h \in \mathcal{H}$ with probability at least $1 - \frac{N^4}{2^k \delta^2} \geq 1 - \frac{1}{2n^3}$,

$$\left| \mathbb{P}_{r \in \{0,1\}^k}[r \in A_{i,l} \wedge h(r) \in A_{l,j}] - \rho(A_{i,l}) \cdot \rho(A_{l,j}) \right| \leq \frac{\rho(A_{i,l}) \cdot \rho(A_{l,j})}{k^2} \leq \frac{\delta}{N^2}$$

So by Union Bound random $h \in \mathcal{H}$, $(\frac{\delta}{N^2}, A, B)$ -good for all A, B with probability at least $\frac{1}{2}$.

Algorithm always returns an h (II)

For a random $h \in \mathcal{H}$ with probability at least $1 - \frac{N^4}{2^k \delta^2} \geq 1 - \frac{1}{2n^3}$,

$$\left| \mathbb{P}_{r \in \{0,1\}^k} [r \in A_{i,l} \wedge h(r) \in A_{l,j}] - \rho(A_{i,l}) \cdot \rho(A_{l,j}) \right| \leq \frac{\rho(A_{i,l}) \cdot \rho(A_{l,j})}{k^2} \leq \frac{\delta}{N^2}$$

So by Union Bound random $h \in \mathcal{H}$, $(\frac{\delta}{N^2}, A, B)$ -good for all A, B with probability at least $\frac{1}{2}$.

$$\begin{aligned} &\left| M^2_{h_1,\ldots,h_{s-1}}[i,j] - M_{h_1,\ldots,h_{s-1},h}[i,j] \right| \\ \leq &\sum_{l \in [N]} \left| \mathbb{P}_{r \in \{0,1\}^k} [r \in A_{i,l} \wedge h(r) \in A_{l,j}] - \rho(A_{i,l}) \cdot \rho(A_{l,j}) \right| \\ \leq &N^2 \cdot \frac{\delta}{N^2} = \delta \end{aligned}$$

Algorithm returns *good* h_s

Claim

If h_s *is returned by the above algorithm, then*

$$\left\| M_{h_1,\ldots,h_s} - M^{2^s} \right\| \leq (2^s - 1)\delta$$

Algorithm returns *good* h_s

Claim

If h_s is returned by the above algorithm, then

$$\left\| M_{h_1,\ldots,h_s} - M^{2^s} \right\| \leq (2^s - 1)\delta$$

We have $\|M^2_{h_1,\ldots,h_{s-1}} - M_{h_1,\ldots,h_s}\| \leq \delta$.

Algorithm returns *good* h_s

Claim

If h_s is returned by the above algorithm, then

$$\left\|M_{h_1,\ldots,h_s} - M^{2^s}\right\| \leq (2^s - 1)\delta$$

We have $\|M^2_{h_1,\ldots,h_{s-1}} - M_{h_1,\ldots,h_s}\| \leq \delta$.

$$\|M_{h_1,\ldots,h_s} - M^{2^s}\| \leq \|M_{h_1,\ldots,h_s} - M^2_{h_1,\ldots,h_{s-1}}\| + \|M^2_{h_1,\ldots,h_{s-1}} - M^{2^s}\|$$

Algorithm returns *good* h_s

Claim

If h_s *is returned by the above algorithm, then*

$$\left\| M_{h_1,\ldots,h_s} - M^{2^s} \right\| \leq (2^s - 1)\delta$$

We have $\|M^2_{h_1,\ldots,h_{s-1}} - M_{h_1,\ldots,h_s}\| \leq \delta$.

$$\|M_{h_1,\ldots,h_s} - M^{2^s}\| \leq \|M_{h_1,\ldots,h_s} - M^2_{h_1,\ldots,h_{s-1}}\| + \|M^2_{h_1,\ldots,h_{s-1}} - M^{2^s}\|$$

$$\begin{aligned}\|M^2_{h_1,\ldots,h_{s-1}} - M^{2^s}\| \leq \|M_{h_1,\ldots,h_{s-1}}\| \cdot \|M_{h_1,\ldots,h_{s-1}} - M^{2^{s-1}}\| \\ + \|M_{h_1,\ldots,h_{s-1}} - M^{2^{s-1}}\| \cdot \|M^{2^{s-1}}\|\end{aligned}$$

Algorithm returns *good* h_s

Claim

If h_s is returned by the above algorithm, then

$$\left\|M_{h_1,\ldots,h_s} - M^{2^s}\right\| \leq (2^s - 1)\delta$$

We have $\|M^2_{h_1,\ldots,h_{s-1}} - M_{h_1,\ldots,h_s}\| \leq \delta$.

$$\|M_{h_1,\ldots,h_s} - M^{2^s}\| \leq \|M_{h_1,\ldots,h_s} - M^2_{h_1,\ldots,h_{s-1}}\| + \|M^2_{h_1,\ldots,h_{s-1}} - M^{2^s}\|$$

$$\begin{aligned}\|M^2_{h_1,\ldots,h_{s-1}} - M^{2^s}\| &\leq \|M_{h_1,\ldots,h_{s-1}}\| \cdot \|M_{h_1,\ldots,h_{s-1}} - M^{2^{s-1}}\| \\ &\quad + \|M_{h_1,\ldots,h_{s-1}} - M^{2^{s-1}}\| \cdot \|M^{2^{s-1}}\| \\ &\leq 1 \cdot (2^{s-1} - 1)\delta + (2^{s-1} - 1)\delta \cdot 1 = (2^s - 2)\delta\end{aligned}$$

Setting Parameters

- Set $k = \log(N) = O(\log n)$. So $t \approx n^c$.
- Set $m = \log t = O(\log n)$.
- Want $(2^m - 1)\delta = \epsilon \implies \delta = \frac{\epsilon}{2^m}$

Final Algorithm

- Compute $h_1, \ldots, h_m$ one by one using the algorithm Find.

Final Algorithm

- Compute $h_1, \ldots, h_m$ one by one using the algorithm FIND.
- Compute $A[i, j] = M_{h_1,\ldots,h_m}[i, j]$ for all $i, j \in [N]$.

Final Algorithm

- Compute $h_1, \ldots, h_m$ one by one using the algorithm Find.
- Compute $A[i, j] = M_{h_1,\ldots,h_m}[i, j]$ for all $i, j \in [N]$.
- Compute $\sum_{j:\text{Accepting state}} A[1, j]$ and accept if this is at least $\frac{2}{3} - \epsilon$ else reject.

Final Algorithm

- Compute $h_1, \ldots, h_m$ one by one using the algorithm Find.
- Compute $A[i, j] = M_{h_1,\ldots,h_m}[i, j]$ for all $i, j \in [N]$.
- Compute $\sum_{j:\text{Accepting state}} A[1, j]$ and accept if this is at least $\frac{2}{3} - \epsilon$ else reject.

Space: The only place where more than $O(\log n)$ space is needed is to store the value of $h_1, \ldots, h_m$. And each h_i can be stored in $O(k) = O(\log n)$ space. So total space used is $O(\log^2 n)$.

Final Algorithm

- Compute $h_1, \ldots, h_m$ one by one using the algorithm Find.
- Compute $A[i, j] = M_{h_1,\ldots,h_m}[i, j]$ for all $i, j \in [N]$.
- Compute $\sum_{j:\text{Accepting state}} A[1, j]$ and accept if this is at least $\frac{2}{3} - \epsilon$ else reject.

Space: The only place where more than $O(\log n)$ space is needed is to store the value of $h_1, \ldots, h_m$. And each h_i can be stored in $O(k) = O(\log n)$ space. So total space used is $O(\log^2 n)$.

Time: For all $s \in [m]$, computing $M_{h_1,\ldots,h_s}[i, j]$ takes $O(2^k)$ times computation of $\mathcal{G}_s(r, h_1, \ldots, h_s)$ for all r and to check if $Q(i; \mathcal{G}_s(r, h_1, \ldots, h_s)) = j$ which takes $O(2^m) \cdot \text{poly}(m)$ time. So Find takes $O(N^2 \cdot 2^{2m} \cdot 2^{m+k})\,\text{poly}(m)$. Hence total time $O(N^2 \cdot 2^{2m} \cdot 2^{m+k})\,\text{poly}(m) \cdot m = \text{poly}(n)$.

Thank You

Lemma

If $\mathcal{H}$ is a universal hash family, then for any $A \subseteq \{0,1\}^k$, $B \subseteq \{0,1\}^m$, $\epsilon > 0$,

$$\mathop{\mathbb{P}}_{h \in \mathcal{H}}[h \text{ is not } (\epsilon, A, B)\text{-good}] \leq \frac{\alpha\beta(1-\beta)}{2^k \epsilon^2}$$

Consider the matrix $M \in \{0,1\}^{2^k \times |\mathcal{H}|}$ where $M[x,h] = 1$ if $h(x) \in B$ and 0 otherwise. For any $x_1 \neq x_2 \in \{0,1\}^k$, $\mathop{\mathbb{E}}_{h \in \mathcal{H}}[M[x_1,h]] = \beta$ and $\mathop{\mathbb{E}}_{h \in \mathcal{H}}[M[x_1,h]M[x_2,h]] = \beta^2$

Appendix ii

$$\mathop{\mathbb{E}}_{h\in\mathcal{H}}\left[\left(\beta - \mathop{\mathbb{E}}_{x\in A}[M[x,h]]\right)^2\right] = \mathop{\mathbb{E}}_{x_1,x_2\in A}\ \mathop{\mathbb{E}}_{h\in\mathcal{H}}[(\beta - M[x_1,h])(\beta - M[x_2,h])]$$

$$= \mathop{\mathbb{E}}_{x_1,x_2\in A}\Big[\beta^2 - \beta \mathop{\mathbb{E}}_{h\in\mathcal{H}}[M[x_1,h]] - \beta \mathop{\mathbb{E}}_{h\in\mathcal{H}}[M[x_1,h]] + \mathop{\mathbb{E}}_{h\in\mathcal{H}}[M[x_1,h]\cdot M[x_2,h]]\Big]$$

$$= \mathop{\mathbb{E}}_{x_1,x_2\in A}\left[\mathop{\mathbb{E}}_{h\in\mathcal{H}}[M[x_1,h]\cdot M[x_2,h]] - \beta^2\right]$$

- For $x_1 \neq x_2 : \mathop{\mathbb{E}}_{h\in\mathcal{H}}[M[x_1,h]\cdot M[x_2,h]] = \beta^2$
- For $x_1 = x_2 : \mathop{\mathbb{E}}_{h\in\mathcal{H}}[M[x_1,h]\cdot M[x_2,h]] = \mathop{\mathbb{E}}_{h\in\mathcal{H}}[M[x_1,h]] = \beta.$

Appendix iii

So,

$$\mathop{\mathbb{E}}_{h\in\mathcal{H}}\left[(\beta - \mathop{\mathbb{E}}_{x\in A}[M[x,h]])^2\right] = \frac{1}{|A|}(\beta-\beta^2) = \frac{\alpha\beta(1-\beta)}{2^k}$$

Now $\mathop{\mathbb{P}}_{x\in\{0,1\}^k}[x\in A \wedge h(x)\in B] = \alpha \mathop{\mathbb{P}}_{x\in A}[h(x)\in B] = \alpha\cdot\mathop{\mathbb{E}}_{x\in A}[M[x,h]]$. So h is not (ϵ, A, B) -good iff

$$\left|\mathop{\mathbb{E}}_{x\in A}[M[x,h]] - \beta\right| \geq \frac{\epsilon}{\alpha}$$

By Markov,

$$\mathop{\mathbb{P}}_{h\in\mathcal{H}}\left[\left|\beta - \mathop{\mathbb{E}}_{x\in A}[M[x,h]]\right| \geq \frac{\epsilon}{\alpha}\right] \leq \frac{\alpha\beta(1-\beta)}{2^k\epsilon^2}$$