Lecture 1: One-way Functions

Introduction

Introduction

- Learn a new Mathematical language

Introduction

- Learn a new Mathematical language
- Encouraged to *speak* this language

Introduction

- Learn a new Mathematical language
- Encouraged to *speak* this language
- Encouraged to *conjecture*

Definition (Algorithm)

An *algorithm* $\mathcal{A}$ is a deterministic Turing machine whose input and output are strings over alphabet $\Sigma = \{0, 1\}$.

Algorithm and Running-time

Definition (Algorithm)

An *algorithm* $\mathcal{A}$ is a deterministic Turing machine whose input and output are strings over alphabet $\Sigma = \{0, 1\}$.

Definition (Running-time)

An algorithm $\mathcal{A}$ is said to run in time $T(n)$ if for all $x \in \{0, 1\}^n$, $\mathcal{A}(x)$ halts within $T(|x|)$ steps. $\mathcal{A}$ runs in polynomial time if there exists a constant c such that $\mathcal{A}$ runs in time $T(n) = n^c$.

Algorithm and Running-time

Definition (Algorithm)

An *algorithm* $\mathcal{A}$ is a deterministic Turing machine whose input and output are strings over alphabet $\Sigma = \{0, 1\}$.

Definition (Running-time)

An algorithm $\mathcal{A}$ is said to run in time $T(n)$ if for all $x \in \{0, 1\}^n$, $\mathcal{A}(x)$ halts within $T(|x|)$ steps. $\mathcal{A}$ runs in polynomial time if there exists a constant c such that $\mathcal{A}$ runs in time $T(n) = n^c$.

An algorithm is *efficient* if it runs in polynomial time.

Algorithm and Running-time

Definition (Algorithm)

An *algorithm* $\mathcal{A}$ is a deterministic Turing machine whose input and output are strings over alphabet $\Sigma = \{0, 1\}$.

Definition (Running-time)

An algorithm $\mathcal{A}$ is said to run in time $T(n)$ if for all $x \in \{0, 1\}^n$, $\mathcal{A}(x)$ halts within $T(|x|)$ steps. $\mathcal{A}$ runs in polynomial time if there exists a constant c such that $\mathcal{A}$ runs in time $T(n) = n^c$.

An algorithm is *efficient* if it runs in polynomial time.

- Think: Why?

Randomized Algorithms

Definition (Randomized (PPT) Algorithm)

A *randomized algorithm*, also called a *probabilitic polynomial-time Turing machine* and abbreviated as PPT, is a Turing machine equipped with an extra randomness tape. Each bit of the randomness tape is uniformly and independently chosen.

Randomized Algorithms

Definition (Randomized (PPT) Algorithm)

A *randomized algorithm*, also called a *probabilitic polynomial-time Turing machine* and abbreviated as PPT, is a Turing machine equipped with an extra randomness tape. Each bit of the randomness tape is uniformly and independently chosen.

- Output is a distribution

Randomized Algorithms

Definition (Randomized (PPT) Algorithm)

A *randomized algorithm*, also called a *probabilitic polynomial-time Turing machine* and abbreviated as PPT, is a Turing machine equipped with an extra randomness tape. Each bit of the randomness tape is uniformly and independently chosen.

- Output is a distribution
- Think: Define using a coin-tossing oracle

Function Computation

Function Computation

Definition (Function Computation)

A *randomized algorithm* $\mathcal{A}$ computes a function $f\colon \{0,1\}^* \to \{0,1\}^*$, if for all $x \in \{0,1\}^*$, $\mathcal{A}$ on input x, outputs $f(x)$ with probability 1. The probability is taken over the random tape of $\mathcal{A}$.

Function Computation

Definition (Function Computation)

A *randomized algorithm* $\mathcal{A}$ computes a function $f\colon \{0,1\}^* \to \{0,1\}^*$, if for all $x \in \{0,1\}^*$, $\mathcal{A}$ on input x, outputs $f(x)$ with probability 1. The probability is taken over the random tape of $\mathcal{A}$.

Without loss of generality, we can restrict to functions with binary output

Function Computation

Definition (Function Computation)

A *randomized algorithm* $\mathcal{A}$ computes a function $f\colon \{0,1\}^* \to \{0,1\}^*$, if for all $x \in \{0,1\}^*$, $\mathcal{A}$ on input x, outputs $f(x)$ with probability 1. The probability is taken over the random tape of $\mathcal{A}$.

Without loss of generality, we can restrict to functions with binary output

- Think: Relax the definition to work with probability $1 - 2^{-|x|}$

Function Computation

Definition (Function Computation)

A *randomized algorithm* $\mathcal{A}$ computes a function $f\colon \{0,1\}^* \to \{0,1\}^*$, if for all $x \in \{0,1\}^*$, $\mathcal{A}$ on input x, outputs $f(x)$ with probability 1. The probability is taken over the random tape of $\mathcal{A}$.

Without loss of generality, we can restrict to functions with binary output

- Think: Relax the definition to work with probability $1 - 2^{-|x|}$
- Think: Amplify an algorithm which is correct only with probability $\frac{1}{2} + \frac{1}{\mathrm{poly}(|x|)}$ into one which is correct with probability $1 - 2^{-|x|}$.

Definition (Non-Uniform PPT)

A *non-uniform probabilistic polynomial-time Turing machine* (abbreviated as n.u. p.p.t.) A is a sequence of probabilistic machines $A = \{A_1, A_2, \ldots\}$ for which there exists a polynomial $d(\cdot)$ such that the description size of $|A_i| < d(i)$ and the running time of A_i is also less than $d(i)$. We write $A(x)$ to denote the distribution obtained by running $A_{|x|}(x)$.

Intuition:

Intuition: A function f is an one-way function if

One-way Functions

Intuition: A function f is an one-way function if

- Easy to compute $f(x)$ given x

One-way Functions

Intuition: A function f is an one-way function if

- Easy to compute $f(x)$ given x
- Difficult to compute x from $f(x)$

One-way Functions

Intuition: A function f is an one-way function if

- Easy to compute $f(x)$ given x: Use language of "Function Computation"
- Difficult to compute x from $f(x)$

Intuition: A function f is an one-way function if

- Easy to compute $f(x)$ given x: Use language of "Function Computation"
- Difficult to compute x from $f(x)$: Use language of "n.u. PPT" and "Function Computation"

One-way Functions

Intuition: A function f is an one-way function if

- Easy to compute $f(x)$ given x: Use language of "Function Computation"
- Difficult to compute x from $f(x)$: Use language of "n.u. PPT" and "Function Computation"
  - May be possible to *partially* recover x

Definition (Strong One-Way Function)

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is a *strong one-way function* if it satisfies the following two conditions:

Definition (Strong One-Way Function)

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is a *strong one-way function* if it satisfies the following two conditions:

1. **Easy to compute.**

Definition (Strong One-Way Function)

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is a *strong one-way function* if it satisfies the following two conditions:

1. **Easy to compute.**

   , and

2. **Hard to invert.**

Definition (Strong One-Way Function)

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is a *strong one-way function* if it satisfies the following two conditions:

1. **Easy to compute.** There is a PPT $\mathcal{C}$ that computes $f(x)$ on all inputs $x \in \{0,1\}^*$, and
2. **Hard to invert.**

Definition (Strong One-Way Function)

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is a *strong one-way function* if it satisfies the following two conditions:

1. **Easy to compute.** There is a PPT $\mathcal{C}$ that computes $f(x)$ on all inputs $x \in \{0,1\}^*$, and
2. **Hard to invert.** For any n.u. PPT adversary $\mathcal{A}$,
   for any input length
   $n \in \mathbb{N}$,

Definition (Strong One-Way Function)

A function $f\colon \{0,1\}^* \rightarrow \{0,1\}^*$ is a *strong one-way function* if it satisfies the following two conditions:

1. **Easy to compute.** There is a PPT $\mathcal{C}$ that computes $f(x)$ on all inputs $x \in \{0,1\}^*$, and
2. **Hard to invert.** For any n.u. PPT adversary $\mathcal{A}$, for any input length $n \in \mathbb{N}$,

Probability of Inversion is small

Definition (Strong One-Way Function)

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is a *strong one-way function* if it satisfies the following two conditions:

1. **Easy to compute.** There is a PPT $\mathcal{C}$ that computes $f(x)$ on all inputs $x \in \{0,1\}^*$, and
2. **Hard to invert.** For any n.u. PPT adversary $\mathcal{A}$, for any input length $n \in \mathbb{N}$,

$$\Pr\Big[\qquad \Big] \leqslant \textit{small}$$

Definition (Strong One-Way Function)

A function $f \colon \{0,1\}^* \to \{0,1\}^*$ is a *strong one-way function* if it satisfies the following two conditions:

1. **Easy to compute.** There is a PPT $\mathcal{C}$ that computes $f(x)$ on all inputs $x \in \{0,1\}^*$, and
2. **Hard to invert.** For any n.u. PPT adversary $\mathcal{A}$,
   for any input length
   $n \in \mathbb{N}$,

$$\Pr\left[x \xleftarrow{\$} \{0,1\}^n \qquad\qquad \right] \leqslant \textit{small}$$

Definition (Strong One-Way Function)

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is a *strong one-way function* if it satisfies the following two conditions:

1. **Easy to compute.** There is a PPT $\mathcal{C}$ that computes $f(x)$ on all inputs $x \in \{0,1\}^*$, and
2. **Hard to invert.** For any n.u. PPT adversary $\mathcal{A}$, for any input length $n \in \mathbb{N}$,

$$\Pr\left[x \xleftarrow{\$} \{0,1\}^n; y \leftarrow f(x) \qquad\qquad \right] \leqslant \textit{small}$$

Definition (Strong One-Way Function)

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is a *strong one-way function* if it satisfies the following two conditions:

1. **Easy to compute.** There is a PPT $\mathcal{C}$ that computes $f(x)$ on all inputs $x \in \{0,1\}^*$, and
2. **Hard to invert.** For any n.u. PPT adversary $\mathcal{A}$, for any input length $n \in \mathbb{N}$,

$$\Pr\left[x \xleftarrow{\$} \{0,1\}^n; y \leftarrow f(x) \colon f(\ \mathcal{A}(\quad y)\ ) = y\right] \leqslant \textit{small}$$

Definition (Strong One-Way Function)

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is a *strong one-way function* if it satisfies the following two conditions:

1. **Easy to compute.** There is a PPT $\mathcal{C}$ that computes $f(x)$ on all inputs $x \in \{0,1\}^*$, and
2. **Hard to invert.** For any n.u. PPT adversary $\mathcal{A}$, there exists a fast decaying function $\nu(\cdot)$ such that for any input length $n \in \mathbb{N}$,

$$\Pr\left[x \xleftarrow{\$} \{0,1\}^n; y \leftarrow f(x) \colon f(\ \mathcal{A}(\quad y)\ ) = y\right] \leqslant \nu(n)$$

One-way Functions

Definition (Strong One-Way Function)

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is a *strong one-way function* if it satisfies the following two conditions:

1. **Easy to compute.** There is a PPT $\mathcal{C}$ that computes $f(x)$ on all inputs $x \in \{0,1\}^*$, and
2. **Hard to invert.** For any n.u. PPT adversary $\mathcal{A}$, there exists a fast decaying function $\nu(\cdot)$ such that for any input length $n \in \mathbb{N}$,

$$\Pr\left[x \xleftarrow{\$} \{0,1\}^n; y \leftarrow f(x) \colon f(\ \mathcal{A}(1^n, y)\ ) = y\right] \leqslant \nu(n)$$

Negligible Function

Definition (Negligible Function)

A function $\nu(n)$ is negligible if for every c, there exists some n_0 such that for all $n > n_0$, $\nu(n) \leqslant \frac{1}{n^c}$.

Negligible Function

Definition (Negligible Function)

A function $\nu(n)$ is negligible if for every c, there exists some n_0 such that for all $n > n_0$, $\nu(n) \leqslant \frac{1}{n^c}$.

1. Negligible function decays faster than all "inverse-polynomial" functions

Negligible Function

Definition (Negligible Function)

A function $\nu(n)$ is negligible if for every c, there exists some n_0 such that for all $n > n_0$, $\nu(n) \leqslant \frac{1}{n^c}$.

1. Negligible function decays faster than all "inverse-polynomial" functions
2. That is, $n^{-\omega(1)}$

Definition (Strong One-Way Function)

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is a *strong one-way function* if it satisfies the following two conditions:

1. **Easy to compute.** There is a PPT $\mathcal{C}$ that computes $f(x)$ on all inputs $x \in \{0,1\}^*$, and
2. **Hard to invert.** For any n.u. PPT adversary $\mathcal{A}$, there exists a negligible function $\nu(\cdot)$ such that for any input length $n \in \mathbb{N}$,

$$\Pr\left[x \xleftarrow{\$} \{0,1\}^n; y \leftarrow f(x)\colon f(\mathcal{A}(1^n, y), y) = y\right] \leqslant \nu(n)$$