B504/I538: Introduction to Cryptography

Spring 2017 • Lecture 3
(2017-01-17)
Assignment 0 is due today; Assignment 1 is out and is due in 2 weeks! (2017-01-31)
Review: Asymptotic notation
Oh s and Omegas, Oh my!

Defn: Let $f: \mathbb{N} \to \mathbb{R}$ and $g: \mathbb{N} \to \mathbb{R}$ be two functions. We say $f \in O(g)$ iff $\exists c > 0$ and $n_c \in \mathbb{N}$, such that $\forall n > n_c$,

$$|f(n)| \leq c \cdot |g(n)|.$$

English: “$f$ is [in] big-O of $g$”

- Intuitively: $f(n)$ grows “no faster” than $g(n)$ as $n \to \infty$
Oh s and Omegas, Oh my!

Defn: Let $f: \mathbb{N} \to \mathbb{R}$ and $g: \mathbb{N} \to \mathbb{R}$ be two functions. We say $f \in \Omega(g)$ iff $\exists c > 0$ and $n_c \in \mathbb{N}$, such that $\forall n > n_c$, $|g(n)| \leq c \cdot |f(n)|$.

English: "$f$ is [in] big-Omega of $g$"

- Intuitively: $f(n)$ grows "no slower" than $g(n)$ as $n \to \infty$
- Equivalently: $f \in \Omega(g)$ iff $g \in O(f)$
Oh's and Omegas, Oh my!

**Defn:** Let $f: \mathbb{N} \rightarrow \mathbb{R}$ and $g: \mathbb{N} \rightarrow \mathbb{R}$ be two functions. We say $f \in o(g)$ iff $\forall c > 0$ and $n_c \in \mathbb{N}$, such that $\forall n > n_c$, $|f(n)| < c \cdot |g(n)|$.

**English:** "$f$ is [in] little-$O$ of $g$"

- Intuitively: $f(n)$ grows "much slower" than $g(n)$ as $n \rightarrow \infty$
Oh's and Omegas, Oh my!

**Defn:** Let $f: \mathbb{N} \to \mathbb{R}$ and $g: \mathbb{N} \to \mathbb{R}$ be two functions. We say $f \in \omega(g)$ iff $\forall c > 0$ and $n_c \in \mathbb{N}$, such that $\forall n > n_c$, $|g(n)| < c \cdot |f(n)|$.

**English:** "$f$ is [in] little-Omega of $g$"

- Intuitively: $f(n)$ grows "much faster" than $g(n)$ as $n \to \infty$
- Equivalently: $f \in \omega(g)$ iff $g \in o(f)$
Oh s and Omegas, Oh my!

**Defn:** Let \( f : \mathbb{N} \to \mathbb{R} \) and \( g : \mathbb{N} \to \mathbb{R} \) be two functions. We say \( f \in \Theta(g) \) iff \( \exists c_1, c_2 > 0 \) and \( n_0 \in \mathbb{N} \), such that for all \( n > n_0 \),

\[
c_1 \cdot |g(n)| \leq |f(n)| \leq c_2 \cdot |g(n)|.
\]

**English:** "\( f \) is \([in] \) big-Theta of \( g \)"

- Intuitively: \( f(n) \) grows "about as fast" as \( g(n) \) as \( n \to \infty \)
- Equivalently: \( f \in \Theta(g) \) iff \( f \in O(g) \) and \( g \in O(f) \)
Oh's and Omegas, Oh my!

- Each of \( O(g) \), \( o(g) \), \( \Theta(g) \), \( \Omega(g) \), and \( \omega(g) \) is an infinite set of real-valued functions.

- \( O \) & \( \Theta \) impose a total order on real-valued functions:
  - \( f \in o(g) \implies f < g \)
  - \( f \in O(g) \implies f \leq g \)
  - \( f \in \Theta(g) \implies f = g \)
  - \( f \in \Omega(g) \implies f \geq g \)
  - \( f \in \omega(g) \implies f > g \)

- Reflexive: \( f \in O(f) \)
- Transitive: \( f \in O(g) \land g \in O(h) \implies f \in O(h) \)
- Antisymmetric: \( f \in O(g) \land g \in O(f) \implies f \in \Theta(g) \)
- Total: \( f \in O(g) \lor g \in O(f) \)

- Common abuse of notation: \( f = O(g) \) or \( f = \Omega(g) \) or \( f = \Theta(g) \), etc.
  - Just keep in mind that “=” means “\( \in \)” not “equals”!
Polynomial functions

**Def**: A function $p : \mathbb{N} \rightarrow \mathbb{R}^+$ is polynomial in $n$ if there exist $c \in \mathbb{N}$ such that $p \in O(n^c)$.

- **Def**: $p$ is superpolynomial in $n$ if it is not polynomial in $n$.
- **Def**: The set of all functions polynomial in $n$ is $\text{poly}(n)$.
  - Common abuse of notation: $p = \text{poly}(n)$ means $p \in \text{poly}(n)$.
- $p$ can be polynomial in $n$ and not be "a polynomial"!
  - Eg.: if $f(n)$ is a polynomial, then $p(n) := f(n) \cdot \log n$ is polynomial in $n$. 

Ryan Henry
Inverse polynomial functions

Defⁿ: A function $p: \mathbb{N} \rightarrow \mathbb{R}^+$ is inverse polynomial in $n$ if $p^{-1} \in \text{poly}(n)$.

• Alt. defⁿ: $p: \mathbb{N} \rightarrow \mathbb{R}^+$ is inverse polynomial if its reciprocal is polynomial

Eg.: $p(n) := \frac{1}{n^{100} \log n}$ is inverse polynomial since $p^{-1}(n) \in O(n^{101})$

Eg.: $q(n) := \frac{1}{n!}$ is not inverse polynomial since $n! \notin \text{poly}(n)$
Probabilistic polynomial time (PPT) algorithms
Turing Machines (TMs)

- A simple, well-defined model of computation
  - Definition is out of scope (try Wikipedia)

Church-Turing Thesis: TMs are universal: anything you can compute in theory, you can compute on a TM!

- Measure running time by number of steps before TM halts
  - Measure robust in that all other "reasonable" models of computation require "polynomially related" number of steps

Including your smartphone and laptop
Turing Machines (TMs)

- All inputs and outputs to a TM are assumed to be finite strings from $\{0, 1\}^*$
  - If $x \in \{0, 1\}^*$, then $|x|$ denotes the length of (number of bits in) $x$

**Defn:** A sequence of TMs $\{TM_n\}_{n \in \mathbb{N}}$ implements an algorithm $A$ if, $\forall x \in \{0, 1\}^n$, on input $x$, $TM_n$ executes each step of $A$ with input $x$. 
Turing Machines (TMs)

- We use the terms *algorithm* and *(sequence of)* TM(s) interchangeably.
  - An algorithm is equivalent to the "most efficient sequence of TMs" that implements it.

**Def:** If $A$ is an algorithm (sequence of TMs), then $A(x)$ denotes the random variable describing the output of $A$ on input $x \in \{0, 1\}^*$. 
Turing Machines (TMs)

**Defn:** Let $t: \mathbb{N} \rightarrow \mathbb{N}$. An algorithm $A$ runs in time $t$ if $\forall x \in \{0, 1\}^*, A$ halts after at most $t(|x|)$ steps on input $x$.

**Note:** running time is proportionate to length of input
- If input is an integer, say $n \in \mathbb{N}$ (expressed in binary), then running time depends on $\lceil \log n \rceil$ rather than on $n$ itself!
Turing Machines (TMs)

Defn: An algorithm $A$ computes $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ if there exists a function $t: \mathbb{N} \rightarrow \mathbb{N}$ such that $A$ runs in time $t$ and $\forall x \in \{0, 1\}^*, A(x) = f(x)$.

- Requirement that $A$ runs in time $t$ is just to ensure that $A$ eventually halts; otherwise, an algorithm that never halts would compute every function!
Polynomial time algorithms

Defn: An algorithm $A$ runs in polynomial time if there exists $p \in \text{poly}(n)$ such that $A$ runs in time $p$.

- Refer to such an algorithm as a polynomial-time algorithm.
Probabilistic algorithms

Defn: An algorithm $A$ is probabilistic if $\exists x \in \{0, 1\}^*$ such that $A(x)$ induces a distribution that is not a point distribution.

• Intuitively, a probabilistic algorithm makes random choices during its execution.

• Probabilistic algorithms can do everything non-probabilistic algorithms can do and possibly more.
Probabilistic polynomial time

Defn: An algorithm $A$ is probabilistic polynomial-time (PPT) if it is probabilistic \textit{and} it runs in polynomial time.

- An algorithm is said to be \textit{efficient} if it is PPT.
- If the sequence of TMs $\{TM_n\}_{n \in \mathbb{N}}$ implementing $A$ satisfies $TM_1 = TM_2 = \cdots$ then $A$ is uniform PPT; otherwise, it is non-uniform PPT (nuPPT).
The negligible, the noticeable, and the overwhelming
Negligible functions

**Def**: A function $\varepsilon: \mathbb{N} \rightarrow \mathbb{R}^+$ is negligible if

$$\forall c > 0, \varepsilon(n) \in O(n^{-c})$$

- **Alt. def**: A function $\varepsilon: \mathbb{N} \rightarrow \mathbb{R}^+$ is negligible if it “vanishes (to zero)” faster than any inverse polynomial

- **Alt. def**: A function $\varepsilon: \mathbb{N} \rightarrow \mathbb{R}^+$ is negligible if $\forall c > 0$, $\exists n_c \in \mathbb{N}$ such that $\varepsilon(n) < n^{-c}$ for all $n > n_c$
Negligible functions

Def^n: A function $\epsilon: \mathbb{N} \rightarrow \mathbb{R}^+$ is negligible if

$$\forall c > 0, \epsilon(n) \in O(n^{-c}).$$

- Def^n: The set of all functions negligible in $n$ is $\text{negl}(n)$
- Common abuse of notation: $\epsilon = \text{negl}(n)$
Negligible functions

Thm (negl + negl = negl): If $\varepsilon_1: \mathbb{N} \rightarrow \mathbb{R}^+$ and $\varepsilon_2: \mathbb{N} \rightarrow \mathbb{R}^+$ are both negligible, then $\varepsilon_1(n) + \varepsilon_2(n)$ is negligible.

Proof: Left as an exercise (see Assignment 1).
Corollary: If $\epsilon: \mathbb{N} \rightarrow \mathbb{R}^+$ is negligible and $\lambda: \mathbb{N} \rightarrow \mathbb{R}^+$ is not negligible, then $\lambda(n) - \epsilon(n)$ is not negligible.

Proof: Define $\mu(n) := \lambda(n) - \epsilon(n)$ and suppose (for a contradiction) that $\mu(n)$ is negligible.

Then, by the preceding theorem, $\mu(n) + \epsilon(n) = \lambda(n)$ would also be negligible.
Negligible functions

Thm \((\text{poly} \times \text{negl} = \text{negl})\): If \(\varepsilon : \mathbb{N} \to \mathbb{R}^+\) is negligible and \(p : \mathbb{N} \to \mathbb{R}^+\) is polynomial, then \(p(n) \cdot \varepsilon(n)\) is negligible.

Proof: Left as an exercise (see Assignment 1). \(\Box\)

Note: This proof is not especially tricky; however, the proof does NOT follow immediately from the fact \(\varepsilon_1(n) + \varepsilon_2(n)\) is negligible—any argument that does not use the fact that \(p(n)\) is a polynomial is incorrect!

Indeed, if \(p(n) = \lceil 1/\varepsilon(n) \rceil\), then \(p(n) \cdot \varepsilon(n) = \sum_{i=1}^{p(n)} \varepsilon(n) \leq 1\) and is clearly not negligible!
Negligible functions

Thm (negl$^c$ = negl): If $\varepsilon: \mathbb{N} \rightarrow \mathbb{R}^+$ is negligible and $c > 0$ is a constant, then $(\varepsilon(n))^c$ is negligible.

Proof: Left as an exercise (see Assignment 1).

Note: This proof is not especially tricky; however, the proof does NOT follow from the lemma on the next slide together with the self-evident yet dubious "fact" that $(\varepsilon(n))^c < \varepsilon(n)$ whenever $\varepsilon(n) < 1$.

Indeed, if $\varepsilon(n) < 1$ and $c < 1$, then $(\varepsilon(n))^c > \varepsilon(n)$. 
Proving a function is negligible

- **Step 1:** Assume $c > 0$ is given
  - do not fix a specific $c$ - keep $c$ is an abstract quantity

- **Step 2:** Set $n_c = f(c)$, for some $f$
  - this step requires all the ingenuity

- **Step 3:** Prove that $\varepsilon(n) < n^{-c}$ for all $n > n_c$

* Okay to assume that $c$ is not "too small"
Lemma: Let \( f: \mathbb{N} \rightarrow \mathbb{R}^+ \) and fix \( c > 0 \) and \( n_c \in \mathbb{N} \) such that \( f(n) < n^{-c} \) for all \( n > n_c \). Then for any \( d \in (0, c) \), we have \( f(n) < n^{-d} \) for all \( n > n_c \).

Q: Why do we care?

A: To prove \( \epsilon \) is negligible, it suffices to consider only "large" values of the constant \( c \) - sometimes the argument for large \( c \) fails for small \( c \).
Lemma: Let $f: \mathbb{N} \rightarrow \mathbb{R}^+$ and fix $c > 0$ and $n_c \in \mathbb{N}$ such that $f(n) < n^{-c}$ for all $n > n_c$. Then for any $d \in (0, c)$, we have $f(n) < n^{-d}$ for all $n > n_c$.

Proof: $n^{-c} \leq n^{-d}$ iff $n^{-c}/n^{-d} \leq 1$ (noting $n \geq 1$)

Rewrite $n^{-c}/n^{-d}$ as $n^{-a}$, where $a = c - d$

Now, $a > 0$ since $c > d$; thus, $1 \leq n^a$ (again noting $n \geq 1$)

It follows that, $n^{-c}/n^{-d} = n^{-a} \leq 1$. □
Example proof: $2^{-\sqrt{n}}$ is negligible

Proof:

Let $c \geq 2$ be given. (assuming $c \geq 2$ justified by preceding lemma)

We must produce an $n_c \in \mathbb{N}$ such that $2^{-\sqrt{n}} < n^{-c}$ for all $n > n_c$.

Taking logs, this is $-\sqrt{n} < -c \log n \iff \sqrt{n}/\log n > c$ for all $n > n_c$.

Set $n_c = \lceil c^4 \rceil$. Since $c \geq 2$, we have $\log n_c \leq n_c^{1/4}$; hence, for all $n > n_c$,

$$\frac{\sqrt{n}}{\log n} \geq \frac{\sqrt{n}}{n^{1/4}} \geq \frac{n^{1/4}}{n^{1/4}} = n^{1/4} > n_c^{1/4}.$$

But $n_c^{1/4} \geq (c^4)^{1/4} = c$, so we just showed $-\sqrt{n}/\log n > c$, as desired. \(\square\)
Example proof: $10^{100} \cdot \varepsilon(n)$ is negligible

Proof:

Let $c > 0$ be given.

By defn: $\exists m_c \in \mathbb{N}$ such that $\varepsilon(n) < n^{-(c+1)}$ for all $n > m_c$.

Set $n_c = \max\{10^{100}, m_c\}$. Then for all $n > n_c$,

$$10^{100} \cdot \varepsilon(n) < 10^{100} \cdot n^{-(c+1)}$$

$$= (10^{100} / n) \cdot n^{-c}$$

$$\leq n^{-c};$$

$$\quad (n_c \geq 10^{100} \Rightarrow (10^{100} / n) \leq 1)$$

hence, we have shown that $10^{100} \cdot \varepsilon(n) < n^{-c}$ for all $n > n_c$. □
Overwhelming functions

**Def**: A function $\kappa: \mathbb{N} \to \mathbb{R}^+$ is overwhelming if $\varepsilon(n) := 1 - \kappa(n)$ is negligible.

- **Alt. def**: A function $\kappa: \mathbb{N} \to \mathbb{R}^+$ is overwhelming if there exists a negligible function $\varepsilon: \mathbb{N} \to \mathbb{R}^+$ such that $\kappa(n) = 1 - \varepsilon(n)$.

- (Like negligible functions), overwhelming functions typically describe probabilities.
Noticeable functions

Def\textsuperscript{n}: A function $\mu: \mathbb{N} \rightarrow \mathbb{R}^+$ is noticeable if

$$\exists c > 0, \mu(n) \in \Omega(n^{-c}).$$

- Alt. def\textsuperscript{n}: A function $\mu: \mathbb{N} \rightarrow \mathbb{R}^+$ is noticeable if some inverse polynomial "vanishes to zero" faster than it.

- Noticeable $\leftrightarrow$ inverse polynomial; however, we typically reserve the label "noticeable" for inverse polynomials that (quickly) tend to 0 as $n \rightarrow \infty$.
If $f$ is not negligible, does that mean it is noticeable?

Q: NO! But why? (See assignment 1)
Distribution ensembles

Defn: An ensemble of probability distributions is an infinite sequence \( \{X_n\}_{n \in \mathbb{N}} \) of random variables indexed by the natural numbers.

- Any PPT algorithm \( A \) induces an ensemble of distributions
  \[ \forall n \in \mathbb{N}, \text{ set } X_n = A(x) \text{ for the } x \in \{0,1\}^n \]
Indistinguishable ensembles

Defn: Two ensembles of distributions \( \{X_i\}_{i \in \mathbb{N}} \) and \( \{Y_i\}_{i \in \mathbb{N}} \) are statistically indistinguishable if there exists a negligible function \( \varepsilon : \mathbb{N} \to \mathbb{R}^+ \) such that for all \( i \in \mathbb{N} \),

$$\sum_{x \in V_i} |\Pr[X_i = x] - \Pr[Y_i = x]| \leq \varepsilon(n)$$

where \( V_i \) is the (combined) range of \( X_i \) and \( Y_i \).
That's all for today, folks!