Outline

Why randomize computation?
Probabilistic Turing Machines
Probabilistic complexity classes
Average case complexity
One-way functions
Pseudorandomness and derandomization

Why do plants randomize their patterns?

Why do ants randomize their paths?

Why do people randomize?
**Randomized algorithm**

**Task**

Decide whether

- $f = 0$ identically, where
- $f(\xi_1, \ldots, \xi_n) \in \mathbb{Q}[\xi_1, \ldots, \xi_n]$ and
- each $\xi_i$ is of degree at most $k$.

**Randomized algorithm**

**Schwartz's Lemma**

If

- for each $f(\xi_1, \ldots, \xi_n) \in \mathbb{Q}[\xi_1, \ldots, \xi_n]$ where
- each $\xi_i$ is of degree at most $k$

then

$$\Pr(f(x_1, \ldots, x_n) = 0 \mid x_i \in \mathbb{Q}[0, N-1]) \leq \frac{kn}{N}$$

**Randomized algorithm**

**Proof of Schwartz's Lemma**

- If $n = 1$ the statement is true since a polynomial of degree $k$ can have at most $k$ zeros.
- Suppose the statement is true for $n - 1$. Then arrange

$$f = f_0 + f_1 x_1 + f_2 x_1 + \cdots + f_N x_1^n$$

where $f_i \in \mathbb{Q}[x_2, x_3, \ldots, x_n]$ and $f_0 \neq 0$.

**Randomized algorithm**

**Proof of Schwartz's Lemma (continued)**

- Then for any randomly chosen $x_i \sim \mathbb{R}$, $i \leq n$, holds

$$\Pr(f(x_1, \ldots, x_n) = 0) = \Pr(f(x_1, \ldots, x_n) = 0) + \Pr(f(x_1, \ldots, x_n) = 0) \cdot \Pr(f(x_2, \ldots, x_n) \neq 0) \cdot \Pr(f(x_3, \ldots, x_n) = 0) \leq \Pr(f(x_2, \ldots, x_n) = 0) + \Pr(f(x_3, \ldots, x_n) = 0)$$

- output $\Pr(f = 0) \geq 1 - 2^{-100}$

**Randomized algorithm**

**Problem**

- sometimes easy:
  - $(x + y)(yz - 8) - y^2 x^2 + 8x^n$
- often exponential in $n$ and $k$
Outline

Why randomize computation?

Probabilistic Turing Machines

Probabilistic complexity classes

Average case complexity

One-way functions

Pseudorandomness and derandomization

Probabilistic Turing Machine

**Def.** A probabilistic Turing machine (PTM) runs on an alphabet in the form $\mathbb{Z}^{\geq 0} \times \{0,1\}$, has a special input tape and strings of $\{0,1\}$ are only on it. Runs on $(x,p)$ for all $x$, where $p$ is a polynomial. This means that at each state $q$ of shape $M_q$, and for each $q \in \mathbb{Z}$, there are exactly 2 transitions.

Computer so far

$$M \xrightarrow{(-)} \ u \xrightarrow{(-)} c$$

$$U, C \in M$$

$$u, C : N \times N \rightarrow N$$

$$u : N \rightarrow N$$

Randomized computer

$$M \xrightarrow{(-)} \ u \xrightarrow{(-)} c$$

$$U, C \in M$$

$$u, C : N \times \{\Delta N\}^\ast \times N \rightarrow N$$

$$u, C : N \rightarrow \{\Delta N\}^\ast$$

$$c : N \rightarrow N$$
Complexity 4: Randomized
Dusko Pavlovic

Why randomize? PPT Classes
Average case One-way Pseudorandomness

Randomized computer

\[ M \xrightarrow{(-)} N \]

\[ (\Delta N)^* \]

\[ \Delta N = \left\{ x : N \rightarrow [0,1] \mid \sum_{n \in N} \mu(n) \leq 1 \wedge \forall n \mu(n) > 0 \right\} \]

Randomized computer

\[ M \xrightarrow{(-)} N \]

\[ (\Delta N)^* \]

\[ \mu(n) = \sum_{x \in N} \mu(x) \]

\[ \Pr(y \leftarrow u_p(x)) = \frac{\# \{ r \mid \tilde{u}(p,x,r) = y \}}{\# \{ r \}} \]

\[ \Delta \]
The defining condition for \( L \in \text{PP} \)

\[ x \in L \implies \Pr(N(x) = \text{"yes"}) > \frac{1}{2} \]

seems to suggest that a PP-language may be such that every machine fails to recognize almost half of its words.

The other classes also seem "lossy" in a similar sense.

The next Proposition shows that that this "lossiness" can be arbitrarily decreased.

**Proposition 1**

Let \( N \in \text{TM} \) and \( L \subseteq \Sigma^* \) satisfy

\[ x \in L \implies \Pr(N(x) = \text{"yes"}) > \frac{1}{2} \]

Then for any \( c > 0 \) there is \( M \in \text{TM} \) such that

\[ x \in L \implies \Pr(M(x) = \text{"yes"}) > 1 - 2^{-|x|^c} \]

**Remark**

Similar statements can be proven for the defining conditions of all probabilistic classes that we defined.

**Corollary**

The constant \( \frac{1}{2} \) in the definitions of the probabilistic classes can be replaced by any number \((\frac{1}{2}, 1), \) or by

\[ 1 - 2^{|x|^c} \]
Interpretation

A probabilistically decidable language may contain some words that particular machines may not recognize in particular runs, or may falsely recognize; but the probability of this can be decreased below any desired threshold by constructing more precise machines.

Outline

Why randomize computation?
Probabilistic Turing Machines
Probabilistic complexity classes
Average case complexity
One-way functions
Pseudorandomness and derandomization

Average case problem

\[ \text{Def. An average-case problem is a pair } (L, \mu) \text{, where} \]
\[ L \subseteq \mathbb{Z}^n \text{ is a language (ordinary poly)} \]
\[ \mu : \mathbb{Z}^n \rightarrow [0, 1] \text{ is a distribution, i.e.} \]
\[ \sum_{x \in \mathbb{Z}^n} \mu(x) = 1. \]

Average case complexity

For arbitrary complexity measure \( C \), the average-case theory is developed by replacing the requirement
\[ \forall x \in L, C(M^N, x) = O(1) \]
by
\[ \sum_{x \in L} \mu(x) C(M^N, x) = O(1) \]
i.e., computations are bounded on average, not all.

Average case reductions

\[ \text{Def. An average-case reduction } R : (L, \mu) \rightarrow (L', \mu') \text{ is a poly-time (or log-space) reduction } \]
\[ R : L \rightarrow L', \text{ such that} \]
\[ \sum_{y \in L'} \mu'(y) = \sum_{x \in L} \mu(x) \mu(R^{-1}(x)). \]
Effective distributions

Definition

A probabilistic distribution \( \mu : \Sigma^* \rightarrow [0, 1] \) is called

- \( P \)-samplable if \( \mu \in P \)
- \( P \)-computable if \( \tilde{\mu} \in P \), where
  \[
  \tilde{\mu}(x) = \sum_{y \subseteq x} \mu(y)
  \]

Proposition 2

- \( P \)-samplable \( \subseteq \) \( P \)-computable
- \( (P \)-computable \( \subseteq \) \( P \)-samplable) \( \implies \) \( P = NP \)

Average polytime nondeterministic languages

Definition

\[ ANP = NP \times P \text{-computable} = \{(L, \mu) \mid L \in NP \land \tilde{\mu} \in P\} \]

Remark

- Our class \( ANP \) is usually denoted \( DNP \) in the literature (following Levin).
- But then when you define \( AP \) (as we do next) you end up with
  \[
  P \overline{NP} = \overline{AP} \overline{DNP}
  \]

Average polytime deterministic languages

Idea

For \( M \in DTM \) we write

\[
\begin{align*}
M \in PTM & \iff \exists d \forall x \in \Sigma^* \forall n. |x| = n \implies \overline{\text{time}_M(x)} \leq n^d \\
M \in APTM^\mu & \iff \exists d \forall x \in \Sigma^* \forall n. |x| = n \implies \int_{x \in L} \overline{\text{time}_M(x)} \leq n^d \\
M \in APTM & \iff \exists d \forall x \in \Sigma^* \forall n. |x| = n \implies \int_{x \in L} \overline{\text{time}_M(x)} \leq 1
\end{align*}
\]

Definition

\[ AP = \{L(M), \mu) \mid M \in APTM \} \]
Average polytime deterministic languages

**Proposition 3**

\[ P = NP \implies AP = ANP \]

Fear of average softness

**Comment**

- Even if \( P \neq NP \) it is still possible that \( AP = NP \)!
- It would mean that
  - hard problems exist
  - but problems are easy on the average, and thus
  - hard puzzles are hard to find.
- Moreover, one-way functions do not exist
  - hard problems exist
  - but we cannot use them :(

Outline

- Why randomize computation?
- Probabilistic Turing Machines
- Probabilistic complexity classes
- Average case complexity
- One-way functions
- Pseudorandomness and derandomization

Modern cryptography
Authentication

Problem (cca 1965)

Users of a shared computer can read the password file.

Solution
- Do not store the passwords $p_1, p_2, p_3, \ldots$
- but the values $f(p_1), f(p_2), f(p_3), \ldots$
- where the function $f : \mathbb{N} \rightarrow \mathbb{N}$ is easy to compute
- but $p_i$ is hard to extract from $f(p_i)$.

One-way functions

Examples

<table>
<thead>
<tr>
<th>easy</th>
<th>hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = a^x$</td>
<td>$x = \log_a{n}$</td>
</tr>
<tr>
<td>$n = x^a$</td>
<td>$x = \sqrt{n}$</td>
</tr>
<tr>
<td>$n = \pi \cdot x$</td>
<td>$x, y</td>
</tr>
</tbody>
</table>

One-way functions

Idea

$n \xrightarrow{f \in \mathbb{P}} x$

$h(x) \in f^{-1}(x) \implies h \notin \mathbb{P}$

One-way functions

Idea

$n \xrightarrow{f \in \mathbb{P}} x$

$f \circ h \circ f = f(h) \implies h \notin \mathbb{P}$
One-way functions

Idea

\[
\begin{align*}
    f \in \text{PT} & \implies \Pr(f(n) \leftarrow f \circ h(n) \mid n \leftarrow \mathbb{N}) \neq 0 \\
    h \in \text{PPT} & \implies \Pr(f^{-1} \circ f(n) \leftarrow h(n) \mid n \leftarrow \mathbb{N}) = 0
\end{align*}
\]

Probabilistic PolyTime (PPT)

Task

Factor out negligible probabilities.

Randomized computers so far

To distinguish the behaviors of polynomially bounded computers \( M \) and \( N \) we may need to sample

\[
\Pr(y \leftarrow M(x)) \quad \text{and} \quad \Pr(y \leftarrow N(x))
\]

exponentially long!

\[
\Pr(y \leftarrow M(x)) = \frac{\#\{r \mid M(x, r) = y\}}{2^{(\ell(x))}}
\]
Indistinguishable random variables

Definition

Sequences of random events \( X, Y : \mathbb{N} \rightarrow \Delta \mathbb{N} \) are indistinguishable if there is a negligible function \( \nu : \mathbb{N} \rightarrow [0, 1] \) such that

\[
\forall n \in \mathbb{N}. \quad \left| \Pr(X_n) - \Pr(Y_n) \right| \leq \nu(n)
\]

When \( X \) and \( Y \) are indistinguishable, we write \( \Pr(X_n) = \Pr(Y_n) \).

Negligible functions

Definition

A function \( \nu : \mathbb{N} \rightarrow [0, 1] \) is negligible if

\[
\forall k \in \mathbb{N} \exists n_0 \in \mathbb{N} \forall n \in \mathbb{N}. \quad n \geq n_0 \implies \nu(n) < \frac{1}{n^k}
\]

Indistinguishable behaviors

Definition

The random maps \( f, g : \mathbb{N} \rightarrow \Delta \mathbb{N} \) are indistinguishable if there is a negligible function \( \nu : \mathbb{N} \rightarrow [0, 1] \) such that

\[
\left| \Pr(y \mapsto f(x)) - \Pr(y \mapsto g(x)) \right| \leq \nu(|x|)
\]

holds for all \( x, y \in \mathbb{N} \).

The indistinguishability relation is written \( f \approx g \).

One-way functions

Lemma

For any function \( f \in \text{PPT} \) the following conditions are equivalent:

\begin{enumerate}
\item \( \forall h \in \text{PPT}. \quad \Pr(f^{-1} \circ f(x) - h(f(x)) \mid x \sim \mathbb{B}^k) = 0 \)
\item \( \forall h \in \text{PPT}. \quad \Pr(f^{-1} \circ f(x) - h(f(x)) \mid x \sim \mathbb{B}^k) = \Pr(f^{-1} \circ f(x) - h(0) \mid x \sim \mathbb{B}^k) \)
\end{enumerate}
One-way functions

**Definition**

A function \( f : \mathbb{N} \rightarrow \mathbb{N} \) is one-way

- \( f \) is in PPT
- \( 3c \cdot |x| \leq |f(x)|^c \)
- the conditions from the preceding Lemma hold.

**Remarks**

- If \( |f(x)|^c < |x| \) for all \( c \), then a machine polynomial in \( |f(x)| \) cannot have time to write a string of length \( |x| \).

**Example**

\[ f : \mathbb{N} \rightarrow \mathbb{N} \]
\[ x \mapsto g^x \mod p \]

is a one-way function, provided that the Discrete Logarithm Assumption (DLA) holds, i.e. that

\[
\Pr\left(x \leftarrow \{p, g, y\} \mid y = g^x \mod p, \quad g, x \in \mathbb{Z}_p^*, \quad p \text{-- Primes}\right) = 0
\]

for all \( t \in \text{PPT} \).

---

**Universal one-way function**

**Proposition 5**

One-way functions exist if and only if the following function \( f : \{0, 1\}^* \rightarrow \{0, 1\}^* \) is one-way

\[ f(x) = p_1(x) :: p_2(x) :: p_3(x) :: \cdots :: p_k(x) \]

where

- \( a :: b \) denotes string concatenation
- \( p_1, p_2, p_3, \ldots \) is a complete list of programs such that
  - \( |a| \leq q(i) \) for some polynomial \( q \in \mathbb{Z}[x] \)
  - \( p_i(x) \) clocks out after \( x^2 \) steps.
Universal one-way function

Lemma
One-way functions exist if and only if there are one-way functions that halt on input $x$ after at most $|x|^2$ steps.

Trapdoor functions

Definition
A trapdoor function is a triple $f = (e, d, t_n)$ such that
- $e, d$ are in PPT
- $\exists c. |x| \leq |e(x)|^c$
- $\forall h \in PPT$. $Pr\left( e^{-1}(e(x)) - h(e(x), u) | x, u \in \{0, 1\}^n \right) \sim 0$
- $\forall h$. $Pr\left( e^{-1}(e(x)) - d(e(x), t_n) | x \in \{0, 1\}^n \right) \sim 1$

Comment
In the literature, trapdoor functions are usually defined as one-way functions that also have a trapdoor. However, the existence of a PPT trapdoor inverter contradicts the statement that there are no PPT inverters, that defines one-way functions.

A trapdoor inverter $d$ comes with an additional argument for the trapdoor $t_n$. As long as the trapdoor is not entered at this argument, there are still no PPT inverters.

Trapdoor functions

Example
Let $(g, \cdot)$ be a large cyclic group generated by $g$ and set
- $f : g \rightarrow \Delta g^2$ such that for all $r \in \mathbb{Z}_2^n$
  $Pr\left( f(m) = \langle g^r, g^r \cdot m \rangle \right) = 2^{-n}$
- $d : g \rightarrow g$ such that
  $d(c_1, c_2) = \frac{c_2}{c_1}$
- where $t \in \mathbb{Z}_2^n$ is the trapdoor.

Summary
Proposition 6
$P \neq NP$
$\uparrow$
$AP \neq ANP$
$\uparrow$
$\exists$ one-way functions
$\uparrow$
$\exists$ trapdoor functions

Summary
Proposition 6
$P = NP$
$\downarrow$
$AP = ANP$
$\downarrow$
$\neg \exists$ one-way functions
$\downarrow$
$\neg \exists$ trapdoor functions
Impagliazzo’s Universes

Algorithmica

\[ P = \text{NP} \]
\[ \Downarrow \]
\[ AP = \text{ANP} \]
\[ \Downarrow \]
\[ \neg \exists \text{one-way functions} \]
\[ \Downarrow \]
\[ \neg \exists \text{trapdoor functions} \]

Hard problems do not exist.

Impagliazzo’s Universes

Heuristica

\[ P \neq \text{NP} \]
\[ \Downarrow \]
\[ AP = \text{ANP} \]
\[ \Downarrow \]
\[ \neg \exists \text{one-way functions} \]
\[ \Downarrow \]
\[ \neg \exists \text{trapdoor functions} \]

Some problems have hard instances, but hard instances are hard to find.

Impagliazzo’s Universes

Pessiland

\[ P = \text{NP} \]
and
\[ AP = \text{ANP} \]
but
\[ \neg \exists \text{one-way functions} \]
\[ \Downarrow \]
\[ \neg \exists \text{trapdoor functions} \]

Some problems are hard on the average, but their solved instances are hard to find.

Impagliazzo’s Universes

Minicrypt

\[ P \neq \text{NP} \]
\[ \Downarrow \]
\[ AP = \text{ANP} \]
and
\[ \exists \text{one-way functions} \]
\[ \Downarrow \]
\[ \neg \exists \text{trapdoor functions} \]

Hard problems with solved instances can be constructed, but the solutions cannot be feasibly encoded.

Impagliazzo’s Universes

Cryptomania

\[ P \neq \text{NP} \]
and
\[ AP = \text{ANP} \]
and
\[ \exists \text{one-way functions} \]
and
\[ \exists \text{trapdoor functions} \]

The world of Public Key Cryptography!
### How strong are one-way functions?

- If $f$ is one-way, then we cannot extract $x$ from $f(x)$.
- Can we extract one bit of information about $x$ from $f(x)$?

### Feasible predicates

**Definition**

A *(feasible) predicate* is a function $B : \{0, 1\}^* \rightarrow \{0, 1\}$ implemented by a *PPT*.
Hardcore predicates

Definition

A predicate \( B : \{0, 1\}^n \rightarrow \{0, 1\} \) is hardcore with respect to the function \( f : \{0, 1\}^* \rightarrow \{0, 1\}^* \) if \( f(x) \) provides no information about the value \( B(x) \), i.e.

\[
\Pr(B(x) - h(f(x)) \mid x \in \{0,1\}^n) \approx \Pr(B(x) - b \mid b \in \{0,1\})
\]

Remark

If a function \( f \) has a hardcore predicate, then \( f \) must be one-way.

Hardcore predicates

Upshot

- \( B(x) \) is not predictable from \( f(x) \) or \( B(f(x)) \).
- \( B(f(x)) \) is not predictable from \( f^{(2)}(x) \) or \( B(f^{(2)}(x)) \)
  - ...
- \( B(f^{(n)}(x)) \) is not predictable from \( B(f^{(n+1)}(x)) \)
  
  and thus

\[
B(f(x)) : B(f^{(2)}(x)) : \cdots : B(f^{(n)}(x)) : \cdots
\]

is right-to-left unpredictable.

Hardcore predicates

Definition

A predicate \( B : \{0, 1\}^* \rightarrow \{0, 1\} \) is hardcore with respect to the function \( f : \{0, 1\}^* \rightarrow \{0, 1\}^* \) if \( f(x) \) provides no information about the value \( B(x) \), i.e.

\[
\Pr(B(x) - h(f(x)) \mid x \in \{0,1\}^n) \approx \Pr(B(x) - b \mid b \in \{0,1\})
\]

or equivalently

\[
\Pr(B(x) - h(f(x)) \mid x \in \{0,1\}^n) \approx \frac{1}{2}
\]

Upshot

- \( B(x) \) is not predictable from \( f(x) \) or \( B(f(x)) \).
Pseudorandom generator (PRG)

Definition

A deterministic polytime function \( g : \{0, 1\}^* \rightarrow \{0, 1\}^\ell \) is a pseudorandom generator if for every \( h \in \text{PPT} \) holds

\[
\Pr\left(1 - h(g(x)) | x \perp | \{0, 1\}^\ell \right) - \Pr\left(1 - h(y) | y \perp | \{0, 1\}^\ell \right)
\]

where \( \ell : \mathbb{N} \rightarrow \mathbb{N} \) is a stretch function.

Proposition 6 (Yao (Barak-Arora 9.11))

If a PPT function outputs bitstrings such that no initial segment provides an advantage for predicting the next bit, then that function is a PRG.

Pseudorandom generator (PRG)

Definition

A deterministic polytime function \( g : \{0, 1\}^* \rightarrow \{0, 1\}^\ell \) is a pseudorandom generator if for every \( h \in \text{PPT} \) holds

\[
\Pr\left(1 - h(g(x)) | x \perp | \{0, 1\}^\ell \right) - \Pr\left(1 - h(y) | y \perp | \{0, 1\}^\ell \right)
\]

where \( \ell : \mathbb{N} \rightarrow \mathbb{N} \) is a stretch function.

Terminology

A stretch function is a map \( \ell : \mathbb{N} \rightarrow \mathbb{N} \) such that \( \ell(n) > n \) for all \( n \).

Pseudorandom generator (PRG)

Proposition 7 (Blum, Micali)

Let \( f : \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell \) be an ONWY and \( B : \{0, 1\}^\ell \rightarrow \{0, 1\} \) its hardcore predicate. Then for every stretch function \( \ell : \mathbb{N} \rightarrow \mathbb{N} \)

\[
g(s) = B(x_1) \cdot B(x_2) \cdots B(x_{\ell(n)})
\]

defines a PRG, where

\[
\begin{align*}
x_0 &= s \\
x_{i+1} &= f(x_i)
\end{align*}
\]

Pseudorandomness

Why randomize?

Randomized

Average case

Classes

PPT

PRGs and ONWYs

Proposition 8

PRGs exist if and only if one-way functions exist.
PRGs and ONWYs

\[ \text{PRG} \implies \text{ONWY} \]
- Let \( g_n : \{0,1\}^n \rightarrow \{0,1\}^{2n} \) be a PRG.
- Set \( f(x_1 \cdots x_n \cdots y_1 \cdots y_n \cdots) = g(x_1 \cdots x_n \cdots x_0) \)

Inverting \( f \) distinguishes \( g(x) \cdot x^T \cdot \{0,1\}^n \) from \( u_{2n} \).

PRGs and ONWYs

\[ \text{ONWY} \implies \text{PRG} \]
- Let \( f : \{0,1\}^* \rightarrow \{0,1\}^* \) be a one-way function
  - \( B : \{0,1\}^* \rightarrow \{0,1\} \) its hardcore predicate
- Set \( g(s) = f(s) : B(s) \)
  - or some version of the Blum-Micali construction.

But we need a one-way function with a hardcore predicate.

PRGs and ONWYs

Lemma (Goldreich-Levin)

Any one-way function \( f : \{0,1\}^* \rightarrow \{0,1\}^* \) induces a one-way function

\[ \begin{align*}
    &\{0,1\}^* \rightarrow \{0,1\}^* \\
    &x \cdot y \rightarrow f(x) \cdot y \\
\end{align*} \]

where \( |x| = |y| \)

with the hardcore

\[ B : \{0,1\}^* \rightarrow \{0,1\} \]

\[ x \cdot y \rightarrow \sum_{i=1}^{k} x_i \cdot y_i \]

PRGs and ONWYs

Moral of \( \text{ONWY} \iff \text{PRG} \)

Trading hardness for randomness
- ONWYs give hardness
- PRGs give randomness

Succinct proof

See Levin’s “Fundamentals of Computing”, Sec. 5.4.
Moral of $\text{ONWY} \iff \text{PRG}$

Trading hardness for randomness
- $\text{ONWY}$s give hardness
- PRGs give randomness
- randomness is expensive
- derandomize: replace random seeds by a PRG.

Derandomizing $\text{BPP}$

Proposition 9 (Yao)
If PRGs exist, then

\[
\text{BPP} = \bigcap_{\epsilon > 0} \text{DTIME}(2^{n\epsilon})
\]

Comments
- When the stretch is exponential, then the PRG must run in time exponential in its random seed.
- If there are languages to support such stretch, then $\text{BPP} = \text{P}$.

Conclusion
- Using one-way functions
- we can generate pseudorandom streams
- and reduce randomized computation to deterministic
- without any distinguishable changes in the output.

Question
- Is randomness obsolete?
- Or are we blind to it?