Lecture 14:
Kolmogorov Complexity
Kolmogorov Complexity: A Universal Theory of Data Compression
The Church-Turing Thesis:

Everyone’s Intuitive Notion of Algorithms = Turing Machines

This is not a theorem – it is a falsifiable scientific hypothesis.

A Universal Theory of Computation
A Universal Theory of *Information*?

Can we quantify how much *information* is contained in a string?

A = 01010101010101010101010101010101

B = 11001001110111010110100101100101

Idea: The more we can “compress” a string, the less “information” it contains....
Thesis: The amount of information in a string $x$ is the length of the *shortest description* of $x$.

How should we “describe” strings?
Some algorithmic process to describe strings...

*Church-Turing Thesis: Use Turing machines with inputs!*

Let $x \in \{0,1\}^*$

**Def:** A *description of $x$* is a string $<M,w>$ such that

$M$ on input $w$ halts with only $x$ on its tape.

**Def:** The *shortest description of $x$*, denoted as $d(x)$, is the lexicographically shortest description of $x$. 
A Specific Pairing Function

**Theorem.** There is a 1-1 computable function 
\[ <,> : \Sigma^* x \Sigma^* \rightarrow \Sigma^* \]  
and computable functions \( \pi_1 \) and \( \pi_2 : \Sigma^* \rightarrow \Sigma^* \) such that:

\[ z = <M,w> \iff \pi_1(z) = M \text{ and } \pi_2(z) = w \]

Define:

\[ <M,w> := 0^{\mid M \mid}1 M w \]

(Example: \( <10110,101> = 0000011011010101 \))

Note that \[ \mid <M,w> \mid = 2\mid M \mid + \mid w \mid + 1 \]
Kolmogorov Complexity (1960’s)

**Def:** The *shortest description of* $x$, denoted as $d(x)$, is the lexicographically shortest string $<M,w>$ such that $M$ on $w$ halts with only $x$ on its tape.

**Def:** The *Kolmogorov complexity of* $x$, $K(x)$, is $|d(x)|$.

**EXAMPLES??**
Let’s first determine some properties of $K$. Examples will fall out of this.
Theorem: There is a fixed $c$ so that for all $x$ in $\{0,1\}^*$

$$K(x) \leq |x| + c$$

“The amount of information in $x$ isn’t much more than $|x|$”

Proof: Define a TM $N$ = “On input w, halt.”

On any string $x$, $N$ on $x$ halts with $x$ on its tape.

Observe that $<N,x>$ is a description of $x$.

Let $c = 2|N| + 1$

Then $K(x) \leq |<N,x>| \leq 2|N| + |x| + 1 \leq |x| + c$
Theorem: There is a fixed $c$ so that for all $n \geq 2$, and all $x \in \{0,1\}^*$, $K(x^n) \leq K(x) + c \log n$.

"The information in $x^n$ isn't much more than that in $x$"

Proof: Define the TM $N = \text{"On input } \langle n, \langle M, w \rangle \rangle, \text{ Let } x = M(w). \text{ Print } x \text{ for } n \text{ times."}"

Let $\langle M, w \rangle$ be the shortest description of $x$. Then $K(x^n) \leq |\langle N, \langle n, \langle M, w \rangle \rangle \rangle|$

$$\leq 2|N| + d \log n + K(x) \leq c \log n + K(x)$$

for some constants $c$ and $d$. 

Repetitive Strings have Low K-Complexity n written in binary.
Theorem: There is a fixed $c$ so that for all $n \geq 2$, and all $x \in \{0,1\}^*$, $K(x^n) \leq K(x) + c \log n$

“ The information in $x^n$ isn’t much more than that in $x$ ”

Recall:

$A = 01010101010101010101010101010101$

For $w = (01)^n$, we have $K(w) \leq K(01) + c \log n$

So for all $n$, $K((01)^n) \leq d + c \log n$ for a fixed $c$, $d$
Does The Computational Model Matter?

Turing machines are one “programming language.” If we use other programming languages, could we get significantly shorter descriptions?

An interpreter is a “semi-computable” function

\[ p : \Sigma^* \rightarrow \Sigma^* \]

Takes programs as input and prints their outputs, but a TM implementing \( p \) may not halt sometimes!

**Definition:** Let \( x \in \{0,1\}^* \). The **shortest description of** \( x \) **under** \( p \), called \( d_p(x) \), is the lexicographically shortest string \( w \) such that \( p(w) = x \).

**Definition:** The **\( K_p \) complexity of** \( x \) is \( K_p(x) := |d_p(x)| \).
Theorem: For every interpreter \( p \), there is a fixed \( c \) so that for all \( x \in \{0,1\}^* \), \( K(x) \leq K_p(x) + c \)

Moral: Using another programming language only changes \( K(x) \) by some additive constant

Proof: Define TM \( M = "On w, simulate p(w) and write its output to tape" \)

Then \( <M,d_p(x)> \) is a description of \( x \).

So \( K(x) \leq |<M,d_p(x)>| \)

\[ \leq 2|M| + K_p(x) + 1 \leq c + K_p(x) \]
There Exist Incompressible Strings

**Theorem:** For all $n$, there is an $x \in \{0,1\}^n$ such that $K(x) \geq n$

“There are incompressible strings of every length”

**Proof:**

(Number of binary strings of length $n) = 2^n$

but

(Number of descriptions of length $< n) \leq (Number of binary strings of length $< n)

= 1 + 2 + 4 + \cdots + 2^{n-1} = 2^n - 1

Therefore, there is at least one $n$-bit string $x$ that does not have a description of length $< n$. 
Random Strings Are Incompressible!

**Theorem:** For all $n$ and $c \geq 1$, 
\[
\Pr_{x \in \{0,1\}^n}[ K(x) \geq n-c ] \geq 1 - 1/2^c
\]

"Most strings are highly incompressible"

**Proof:** 
(Number of binary strings of length $n$) = $2^n$ but (Number of descriptions of length $< n-c$) 
\[
\leq (\text{Number of binary strings of length } < n-c) = 2^{n-c} - 1
\]
Hence the probability that a *random* $x$ satisfies 
\[
K(x) < n-c
\]
is at most $(2^{n-c} - 1)/2^n < 1/2^c$. 
Kolmogorov Complexity: Try it!

Give short algorithms for generating the strings:

1. 01000110110000010100111001011101110000
2. 1235813213455891442333776109871597
3. 126241207205040403203628803628800
Kolmogorov Complexity: Try it!

Give short algorithms for generating the strings:

1. 01000110110000010100111001011101110000

2. 1235813213455891442333776109871597

3. 126241207205040403203628803628800
Kolmogorov Complexity: Try it!

Give short algorithms for generating the strings:

1. 0100011011000001010011100101101110000

2. 1235813213455891442333776109871597

3. 126241207205040403203628803628800
Kolmogorov Complexity: Try it!

Give short algorithms for generating the strings:

1. 01000110110000001010011100101110110000
2. 1235813213455891442333776109871597
3. 126241207205040403203628803628800

This looks hard to determine in general. Why?
Computing Compressibility?

Can an algorithm perform optimal compression?
Can algorithms tell us if a given string is compressible?

COMPRESS = \{ (x,c) \mid x \text{ string}, c \text{ integer, } K(x) \leq c \}

**Theorem:** COMPRESS is undecidable!

Idea: If COMPRESS were decidable, we could use its decider to design an algorithm printing an incompressible string of a given length $n$.
But then, such a string could be succinctly described, by providing the algorithm code and $n$ in binary!

**Berry Paradox:** “The smallest integer that cannot be defined in less than thirteen words.”
Computing Compressibility?

**COMPRESS** = \{ (x,c) | x string, c integer, K(x) ≤ c \}

**Theorem:** **COMPRESS** is undecidable!

**Proof:** Suppose it’s decidable. Consider the TM:

M = “On input x ∈ \{0,1\}^*,
    For all y ∈ \{0,1\}^* in lexicographical order,
    If (y, 2^{|x|}) \not\in **COMPRESS** then print y and halt.”

M on x prints the lex. first string y’ with K(y’) > 2^{|x|}.

<M,x> is a description of y’, and |<M,x>| ≤ d + |x| for some constant d

So 2^{|x|} < K(y’) ≤ d + |x|. **CONTRADICTION** for long x!
Yet Another Proof that $A_{TM}$ is Undecidable!

$\text{COMPRESS} = \{(x, c) \mid K(x) \leq c\}$

**Theorem:** $A_{TM}$ is undecidable.

**Proof:** Mapping reduction from $\text{COMPRESS}$ to $A_{TM}$. Given a pair $(x, c)$, our reduction constructs a TM:

$M_{x,c} = \text{On input } w,$

*For all pairs } <M',w'> \text{ with } |<M',w'>| \leq c,$

simulate each $M'$ on $w'$ in parallel.

*If some $M'$ ever halts and prints $x$, then accept.*

$(x, c) \in \text{COMPRESS} \iff K(x) \leq c \iff M_{x,c} \text{ accepts } \varepsilon$
Proving Theorems With K-Complexity

**Theorem:** $L = \{xx \mid x \in \{0, 1\}^*\}$ is not regular.

**Proof:** Suppose $L$ is recognized by a DFA $D$. Let $n \geq 0$ and choose an $x \in \{0, 1\}^n$ such that $K(x) \geq n$. Let $q_x$ be the state of $D$ reached after reading in $x$.

Define TM $M$ that on input $(D, q, n)$ does the following:
- Find some path $P$ in $D$ of $n$ edges from state $q$ to some final state (if no path, reject).
- Print the $n$-bit string along path $P$, and halt.

Claim: The string $<M, (D, q_x, n)>$ is a description of $x$!

So $n \leq K(x) \leq |<M, (D, q_x, n)>| \leq O(\log n)$

CONTRADICTION for large $n$
More on Interesting Formal Systems

A formal system $\mathcal{F}$ is *interesting* if:

1. Any mathematical statement about computation can also be effectively described within $\mathcal{F}$.
   
   For all strings $x$ and integers $c$, there is a $S_{x,c}$ in $\mathcal{F}$ that is equivalent to "$K(x) \geq c$"

2. Proofs are convincing: it should be possible to check that a proof of a theorem is correct.
   
   This set is decidable: $\{ (S,P) \mid P$ is a proof of $S$ in $\mathcal{F} \}$
The Unprovable Truth About K-Complexity

**Theorem:** For every interesting consistent \( \mathcal{F} \), there is a \( t \) s.t. for all \( x \), “\( K(x) > t \)” is unprovable in \( \mathcal{F} \).

**Proof:** Define a Turing machine \( M \) as follows:

\[
M(y) := \text{Search over all strings } x \text{ and proofs } P \text{ for a proof } P \text{ in } \mathcal{F} \text{ of “} K(x) > 2^{|y|} \text{”}. \text{Output } x \text{ if found}
\]

If \( M(y) \) halts, it prints some \( x \). Then for some \( c \),

\[
K(x') = K(<M,y>) \leq c + |y|
\]

Therefore “\( K(x) \leq c + |y| \)” has a proof in \( \mathcal{F} \).

But “\( K(x) > 2^{|y|} \)” also has a proof \( P \) in \( \mathcal{F} \).

For large enough \( |y| \), have proof of “\( K(x') > c + |y| \)” and its negation! Therefore \( M(y) \) does not halt!
Theorem: For every interesting consistent $\mathcal{F}$, there is a $t$ s.t. for all $x$, “$K(x) > t$” is unprovable in $\mathcal{F}$.

For a randomly chosen $x$ of length $t+100$, “$K(x) > t$” is true with probability at least $1-\frac{1}{2^{100}}$.

We can *randomly generate* true statements in $\mathcal{F}$ which have no proof in $\mathcal{F}$, with high probability!

For every interesting formal system $\mathcal{F}$ there is always some finite integer (say, $t=10000$) so that you’ll never be able to prove in $\mathcal{F}$ that a random 20000-bit string requires a 10000-bit program!
Next Episode:
Complexity Theory!