15-453

FORMAL LANGUAGES, AUTOMATA AND COMPUTABILITY

KOLMOGOROV-CHAITIN (descriptive) COMPLEXITY

TUESDAY, MAR 18

CAN WE QUANTIFY HOW MUCH INFORMATION IS IN A STRING?

A = 01010101010101010101010101010101

B = 110010011101110101101001011001011

Idea: The more we can "compress" a string, the less "information" it contains….

INFORMATION AS DESCRIPTION

INFORMATION IN A STRING:
SHORTEST DESCRIPTION OF THE STRING

How can we "describe" strings?

Turing machines with inputs!

KOLMOGOROV COMPLEXITY

Definition: Let **x in {0,1}***. The **shortest description of x**, denoted as **d(x)**, is the **lexicographically shortest string <M,w>** s.t. M(w) halts with x on tape.

Definition: The **Kolmogorov complexity of x**, denoted as **K(x)**, is **|d(x)|.**

How to code <M,w>?

Assume w in {0,1}* and we have a binary encoding of M

THE PAIRING FUNCTION

Theorem. There is a 1-1 and onto computable function <,>: $\Sigma^* \times \Sigma^* \to \Sigma^*$ and computable functions π_1 and π_2 : $\Sigma^* \to \Sigma^*$ such that:

$z = \langle M,w\rangle \Rightarrow \pi_1(z) = M$ and $\pi_2(z) = w$

Let $Z(x_1 x_2 \ldots x_k) = 0\, x_1\, 0\, x_2 \ldots 0\, x_k\, 1$
Then:

$\langle M,w\rangle := Z(M)\, w$

(Example: <10110,101> = 01000101001101)

Note that $|\langle M,w\rangle| = 2|M| + |w| + 1$

A BETTER PAIRING FUNCTION

Let b(n) be the binary encoding of n
Again let $Z(x_1\ x_2 \ldots x_k) = 0\ x_1\ 0\ x_2 \ldots 0\ x_k\ 1$

<M,w> := Z(b(|M|)) M w

Example: Let <M,w> = <10110,101>
So, b(|10110|) = 101
So, <10110,101> = 010001110110101

We can still decode 10110 and 101 from this!

Now, |<M,w>| = 2 log(|M|) + |M| + |w| + 1

KOLMOGOROV COMPLEXITY

Definition: Let **x in {0,1}***. The **shortest description of x**, denoted as **d(x)**, is the **lexicographically shortest string <M,w>** s.t. M(w) halts with x on tape.

Definition: The **Kolmogorov complexity of x**, denoted as **K(x)**, is **|d(x)|.**

EXAMPLES??

Let's start by figuring out some properties of **K**.
Examples will fall out of this.

KOLMOGOROV COMPLEXITY

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 M = "On input w, halt."
On any string x, M(x) halts with x on its tape!
This implies
$K(x) \leq |\langle M,x\rangle| \leq 2|M| + |x| + 1 \leq |x| + c$
(Note: M is fixed for all x. So |M| is constant)

REPETITIVE STRINGS

Theorem: There is a fixed **c** so that for all **x in {0,1}***,
$K(xx) \leq K(x) + c$

"The information in **xx** isn't much more than that in **x**"

Proof: Let N = "On <M,w>, let M(w) = s. Print ss."
Let <M,w'> be the shortest description of x.
Then <N,<M,w'>> is a description of xx
Therefore
$K(xx) \leq |\langle N,\langle M,w'\rangle\rangle| \leq 2|N| + K(x) + 1 \leq K(x) + c$

REPETITIVE STRINGS

Corollary: There is a fixed **c** so that for all **n**,
and all **$x \in \{0,1\}^*$**,
$K(x^n) \leq K(x) + c \log_2 n$

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

Proof:
An intuitive way to see this:
Define M: "On <x, n>, print x for n times".
Now take <M,<x,n>> as a description of x^n.
In binary, n takes O(log n) bits to write down, so we have K(x) + O(log n) as an upper bound on $K(x^n)$.

REPETITIVE STRINGS

Corollary: There is a fixed **c** so that for all **n**,
and all **$x \in \{0,1\}^*$**,
$K(x^n) \leq K(x) + c \log_2 n$

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

REPETITIVE STRINGS

Corollary: There is a fixed **c** so that for all **n**, and all $x \in \{0,1\}^*$,

$$K(x^n) \leq K(x) + c \log_2 n$$

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

Recall:
A = 0101010101010101010101010101010101

For $w = (01)^n$, $K(w) \leq K(01) + c \log_2 n$

CONCATENATION of STRINGS

Theorem: There is a fixed **c** so that for all **x , y in {0,1}***,

$$K(xy) \leq 2K(x) + K(y) + c$$

Better: $K(xy) \leq 2 \log K(x) + K(x) + K(y) + c$

DOES THE LANGUAGE MATTER?

Turing machines are one programming language. If we use other programming languages, can we get shorter descriptions?

An interpreter is a (partial) computable function $p : \Sigma^* \to \Sigma^*$

Takes programs as input, and prints their outputs

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

Definition: $K_p(x) = |d_p(x)|$.

DOES THE LANGUAGE MATTER?

Theorem: For every interpreter **p**, there is a fixed **c** so that for all $x \in \{0,1\}^*$,

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

Using any other programming language would only change K(x) by some constant

Proof: Define M_p = **"On input w, output p(w)"**

Then $\langle M_p, d_p(x)\rangle$ is a description of **x**, and

$$K(x) \leq |\langle M_p, d_p(x)\rangle| \leq 2|M_p| + K_p(x) + 1 \leq K_p(x) + c$$

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

(Number of descriptions of length < n)
≤ (Number of binary strings of length < n)
= $2^n - 1$.

Therefore: there's at least one n-bit string that doesn't have a description of length < n

INCOMPRESSIBLE STRINGS

Theorem: For all **n** and **c**,

$$\Pr_{x \in \{0,1\}^n}[\, K(x) \geq n-c \,] \geq 1 - 1/2^c$$

"Most strings are fairly incompressible"

Proof: (Number of binary strings of length n) = 2^n

(Number of descriptions of length < n-c)
≤ (Number of binary strings of length < n-c)
= $2^{n-c} - 1$.

So the probability that a random x has K(x) < n-c is at most $(2^{n-c} - 1)/2^n < 1/2^c$.

A QUIZ (NOT REALLY)

Give short algorithms for generating:

1. **01000110110000010100111001011101110000**
2. **123581321345589144233377610**
3. **126241207205040403203628800**

This seems hard in general. Why?
We'll give a formal answer in just one moment…

DETERMINING COMPRESSIBILITY

Can an algorithm help us compress strings?
Can an algorithm tell us when a string is compressible?

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

Theorem: COMPRESS is undecidable!

Intuition: If decidable, we can design an algorithm that prints the "first incompressible string of length **n**"
*But such a string could be described succinctly, by giving the algorithm, and **n** in binary!*

"The first string whose shortest description cannot be written in less than fifteen words."

DETERMINING COMPRESSIBILITY

COMPRESS = $\{(x,n) \mid K(x) \le n\}$

Theorem: COMPRESS is undecidable!

Proof:
M = "On input $x \in \{0,1\}^*$,
Interpret x as integer n. ($|x| \le \log n$)
Find first $y \in \{0,1\}^*$ in lexicographical order,
s.t. $(y,n) \notin$ COMPRESS, then print y and halt."

M(x) prints the first string y^* with $K(y^*) > n$.
Thus <M,x> describes y^*, and $|\langle M,x\rangle| \le c + \log n$
So $n < K(y^*) \le c + \log n$. CONTRADICTION!

DETERMINING COMPRESSIBILITY

Theorem: K is not computable

Proof:
M = "On input $x \in \{0,1\}^*$,
Interpret x as integer n. ($|x| \le \log n$)
Find first $y \in \{0,1\}^*$ in lexicographical order,
s. t. $(K(y) > n)$, then print y and halt."

M(x) prints the first string y^* with $K(y^*) > n$.
Thus <M,x> describes y^*, and $|\langle M,x\rangle| \le c + \log n$
So $n < K(y^*) \le c + \log n$. CONTRADICTION!

SO WHAT CAN YOU DO WITH THIS?

Many results in mathematics can be proved very simply using incompressibility.

Theorem: There are infinitely many primes.

IDEA: Finitely many primes $\Rightarrow$ can compress everything!

Proof: Suppose not. Let $p_1, \dots, p_k$ be the primes.
Let x be incompressible. Think of n = x as integer.
Then there are e_i s.t.

$$n = p_1^{e1} \dots p_k^{ek}$$

For all i, $e_i \le \log n$, so $|e_i| \le \log \log n$
Can describe n (and x) with $k \log \log n + c$ bits!
But x was incompressible… CONTRADICTION!

WWW.FLAC.WS

Read Chapter 7.1 for next time