Here are some notes about the Kolmogorov complexity material that we've covered toward the end of the course. A good reference is the textbook *Introduction to the Theory of Computation* by Michael Sipser.

1. Introduction

All strings in these notes are taken over the alphabet $\{0, 1\}$. Consider these two strings:

$$01010101010101010101010101010101,$$

$$01100001010111101101001000001100.$$

Both have length 32, but the latter seems more complicated, because the former has an obvious pattern — "repeat 01 16 times". Our goal is to develop a theory of information complexity that measures how much information is actually contained in a string. The basic idea is to come up with a system for describing (compressing) strings, and then to define the information content of a given string x to be the length of its shortest description.

Our default description system will describe strings x as strings $M\#w$, where M is a Turing machine, w is an input to M, and M outputs x when it is given the input w. In our initial example, the string $x = 01010101010101010101010101010101$ could be described as $M\#w$ where $w = 01$ and M implements "repeat 16 times". The special symbol # is there just to delimit M and w clearly.

Since we are using the alphabet $\{0, 1\}$, we must agree on an encoding of $M\#w$ as bits. Earlier in the course, when we studied diagonalization, we agreed on a standard encoding of M and w into bits. It remains to encode #. Here is one way. Let's agree that # is the string 01. Define a function double() that, given any string, returns the string with each bit doubled up to two bits. For example, double(00101) = 0000110011. Then we can unambiguously encode $M\#w$ into bits as double$(M)01w$. This is a bit wasteful, since it doubles the number of bits needed to store M. One can imagine more sophisticated description systems that yield shorter descriptions, but we don't care much.

Before we state the key definitions, recall that lexicographic order is alphabetical order, with short strings preceding longer strings. The lexicographic order of all bitstrings goes

$$\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, \ldots.$$

Definition 1.1. *Let x be any string. The **minimal description** $d(x)$ is the lexicographically minimal string $M\#w$ such that M is a Turing machine, w is an input to M, and when M is given w as input it halts with x on its tape. The **Kolmogorov complexity** $K(x)$ is the length $|d(x)|$ of the minimal description.*

2. Basic Results

Intuitively, because $d(x)$ is the minimal description of x, you should think of it as the compressed version of x. For this notion of compression to be useful, it should compress strings

down to smaller strings. That is, one would hope that $|d(x)| < |x|$ for all x. However, this is not achievable. (See Lemma 3.3.) Some strings x have so little "pattern" that any attempt to describe the pattern of x takes as much space as simply laying out x itself. To handle such strings in our compression system, we could agree on a flag that indicates "the following string is being given in raw form; don't do anything to it". Then any string x can be described in $|x| + |\text{flag}|$ bits — or less, if x can really be compressed. This is the basic idea behind the following theorem.

Theorem 2.1. *There exists a constant c such that for all strings x*

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

Proof. Let M be a Turing machine that immediately halts. That is, M's output is always identical to its input; M computes the identity function. Let $c = |M| + |\#|$. Then for any string x, $M\#x$ is a description of x. Therefore $K(x) \leq |M\#x| = |x| + c$. □

For any string x, the complexity of xx shouldn't be much more than the complexity of x, because you can describe xx by describing x and then saying "repeat it". This is the basic idea behind the following theorem.

Theorem 2.2. *There exists a constant c such that for all strings x*

$$K(xx) \leq K(x) + c.$$

Proof. Let M be a Turing machine that, when given an input of the form $N\#w$ (where N is a Turing machine and w is an input to N):

(1) Runs N on W to produce an output s.
(2) Repeats s on its tape, to produce a final output ss.

Let $c = |M| + |\#|$. Then for any string x, $M\#d(x)$ is a description of xx. Therefore $K(xx) \leq |M\#d(x)| = |d(x)| + c$. □

Similarly, describing xy shouldn't be much harder than describing x and describing y. The following theorem expresses this idea, but a factor of 2 creeps in due to the idiosyncrasies of our encoding system. This factor can be reduced by a more clever choice of encoding, but it can never be reduced to 1. (That's an exercise for later.)

Theorem 2.3. *There exists a constant c such that for all strings x and y*

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

Proof. For any string w let double(w) be the doubling function defined above. Then we can describe xy as $M\#\text{double}(d(x))01d(y)$ for a suitable Turing machine M. Namely, M scans its input $\text{double}(d(x))01d(y)$ until it finds the special code 01 after an even number of bits. The bits preceding 01 are undoubled to produce $d(x)$, which is decoded to produce x. The bits after

01 are taken to be $d(y)$, which is decoded to produce y. Finally x and y are concatenated on M's tape and M halts. Let $c = |M| + |\#| + |01|$. Then

$$\begin{aligned} K(xy) &\leq |M\#\text{double}(d(x))01d(y)| \\ &= 2|d(x)| + |d(y)| + |M| + |\#| + |01| \\ &= 2K(x) + K(y) + c. \end{aligned}$$

□

3. Incompressibility

Now we make the connection between information complexity and compression explicit.

Definition 3.1. *A string x is **compressible by** c if $K(x) \leq |x| - c$. We say that x is **incompressible** if it is not compressible by 1.*

You would expect that $d(x)$, being the minimal description of x, should be incompressible. For if it were compressible, then there would exist a description of $d(x)$ shorter than $d(x)$ itself, and this description of $d(x)$ would yield a description of x that might be shorter than $d(x)$. This is correct, except that it ignores the inevitable overhead of the encoding system. The following theorem makes it all precise.

Theorem 3.2. *There exists a constant b such that for all strings x, $d(x)$ is incompressible by b.*

Proof. Let M be a Turing machine that on input $N\#w$ does the following steps.

(1) Run N on w.
(2) If the output of N is not of the form $P\#y$, where P is a Turing machine and y is an input for it, then reject.
(3) If the output of N is of the form $P\#y$, then run P on y and halt with that output.

Let $b = |M| + |\#| + 1$. Suppose (for the sake of contradiction) that x is a string such that $d(x)$ is compressible by b. Thus $|d(d(x))| \leq |d(x)| - b$. But $M\#d(d(x))$ is a description of x, and its length is

$$|M| + |\#| + |d(d(x))| \leq (b-1) + |d(x)| - b = |d(x)| - 1.$$

No description of x can have length less than $d(x)$; this contradiction proves that the string x cannot exist, and that proves the theorem. □

Earlier we mentioned that you can't hope to compress every string. The following result explains this in an extremely simple way.

Lemma 3.3. *For every $n \geq 0$ there exists an incompressible string x of length n.*

Proof. Let $n \geq 0$. There are 2^n strings of length n. There are only $2^n - 1$ strings of length less than n, which can describe at most $2^n - 1$ strings. Hence at least one string of length n cannot be described by any shorter string. □

If the preceding result disappointed you, steel yourself.

Theorem 3.4. *The Kolmogorov complexity K is not computable.*

Proof. Suppose (for the sake of contradiction) that K is computable. Let M be a total Turing machine that on input x halts with $K(x)$ on its tape. Use M to construct a total Turing machine N that, on input n (regarded as a base-2 integer) outputs some string x satisfying $K(x) \geq n$. (N tries all strings in lexicographic order, using M to compute K for each, until it finds x with $K(x) \geq n$. The preceding lemma guarantees that this will happen for some x with $|x| \leq n$.)

Let U be a universal Turing machine that on input $L\#w$ simulates L on w and halts with tape equal to whatever L's final tape is. Let m be any integer such that

$$m - \log_2 m > |U| + |\#| + |N| + |\#|.$$

Finally, let $x = N(m)$. Then $U\#N\#m$ is a description of x. Its length is

$$|U\#N\#m| = |U| + |\#| + |N| + |\#| + \log_2 m,$$

because the integer m is represented in base 2. By the definition of m, this length is less than m. Therefore $K(x) < m$. But the definition of N guarantees that $x = N(m)$ has $K(x) \geq m$. From this contradiction we conclude that our initial assumption, that K is computable, was false. □

4. Our Definition Of Complexity Is Equivalent To Any Other

We have defined the information complexity $K(x)$ of a bitstring x based on an encoding of x into a Turing machine M and an input w to that Turing machine. It is natural to ask whether a different notion of information complexity would yield different results. The following theorem answers "no"; our default definition yields similar theorems to any other — just with different constants.

To see this, let $p : \{0,1\}^* \to \{0,1\}^*$ be any computable function; this will be our decoder. To say that it is computable is to say that there exists a total Turing machine M_p that, given input y, halts with $p(y)$ on its tape. Let $d_p(x)$ be the lexicographically minimal string such that $p(d_p(x)) = x$; that is, $d_p(x)$ is the minimal description of x, with respect to the description system p. Let $K_p(x) = |d_p(x)|$; this is the informational complexity of x with respect to the description system p. (For example, our default encoding system amounts to the universal Turing machine U, that on input $y = M\#w$ simulates M on w.)

Theorem 4.1. *For any computable $p : \{0,1\}^* \to \{0,1\}^*$ there exists a constant c such that for all strings x, $K(x) \leq K_p(x) + c$.*

Proof. Let p be given. Let M_p be a Turing machine that implements p. Let $c = |M_p| + |\#|$. Then $M_p\#d_p(x)$ is a description of x. Its length is $c + K_p(x)$. Thus $K(x) \leq K_p(x) + c$. □

5. Randomness And Incompressibility

Definition 5.1. *A **property** of strings over Σ is a function $f : \Sigma^* \to \{\mathrm{T}, \mathrm{F}\}$. A property f **holds for almost all strings** if*

$$\lim_{n\to\infty} \frac{\#\{x : |x| = n, f(x) = \mathrm{F}\}}{\#\{x : |x| = n\}} = 0.$$

The following mathematical lemma shows that we can replace "=" with "≤" in the above definition. Sipser uses this fact without proof. You may want to skip the proof on a first reading.

Lemma 5.2. *Let f be a property that holds for almost all strings. Then*

$$\lim_{n\to\infty} \frac{\#\{x : |x| \le n, f(x) = \mathrm{F}\}}{\#\{x : |x| \le n\}} = 0.$$

Proof. Let $\epsilon > 0$. We wish to show that there exists N such that for all $n \ge N$

$$\frac{\#\{x : |x| \le n, f(x) = \mathrm{F}\}}{\#\{x : |x| \le n\}} < \epsilon.$$

For the sake of brevity, let $L_n = \#\{x : |x| = n, f(x) = \mathrm{F}\}$. Because f holds for almost all strings, there exists an M such that for all $n > M$,

$$\frac{\#\{x : |x| = n, f(x) = \mathrm{F}\}}{\#\{x : |x| = n\}} < \frac{\epsilon}{2}.$$

That is, $L_n < \frac{\epsilon}{2} 2^n$ for all $n > M$. Pick N large enough so that

$$\sum_{i=0}^{M} L_i < \frac{\epsilon}{2}\left(2^{N+1} - 1\right).$$

Then for all $n \ge N$

$$\begin{aligned}
\#\{x : |x| \le n, f(x) = \mathrm{F}\} &= \sum_{i=0}^{M} L_i + \sum_{i=M+1}^{n} L_i \\
&< \sum_{i=0}^{M} L_i + \sum_{i=M+1}^{n} \frac{\epsilon}{2} 2^i \\
&< \frac{\epsilon}{2}\left(2^{N+1} - 1\right) + \frac{\epsilon}{2}(2^{n+1} - 1) \\
&\le \epsilon\left(2^{n+1} - 1\right) \\
&= \epsilon\, \#\{|x| \le n\}.
\end{aligned}$$

This proves the lemma. □

Intuitively, a string generated at random should have no pattern and should not be compressible. The following theorem makes this intuition precise.

Theorem 5.3. *Let f be a computable property that holds for almost all strings. Let $b > 0$. Then $f(x) = \mathrm{F}$ for only finitely many strings that are incompressible by b.*

Proof. If f is false on only finitely many strings, then the theorem is obviously true. Henceforth assume that f is false on infinitely many strings. Denote these strings $s_0, s_1, s_2, \ldots$ in lexicographic order.

For any string x in the sequence $s_0, s_1, s_2, \ldots$, let i_x be its index in the list. That is, i_x is the unique number such that $s_{i_x} = x$. Let M be a Turing machine that on input i, regarded as a base-2 integer, outputs s_i. Then $M\#i_x$ is a description of x.

Fix $b > 0$. By the lemma, there exists a large N so that for all $n \geq N$

$$\frac{\#\{x : |x| \leq n, f(x) = \mathrm{F}\}}{\#\{x : |x| \leq n\}} < \frac{1}{2^{b+|M|+|\#|+1}}.$$

Using the fact that $\#\{x : |x| \leq n\} = 2^{n+1} - 1$, we have

$$\#\{x : |x| \leq n, f(x) = \mathrm{F}\} < \frac{2^{n+1}}{2^{b+|M|+|\#|+1}} = 2^{n-b-|M|-|\#|}.$$

If x is any string of length $n \geq N$ such that $f(x) = \mathrm{F}$, then $i_x < 2^{n-b-|M|-|\#|}$ and $|i_x| \leq n - b - |M| - |\#|$. This implies that

$$K(x) \leq |M\#i_x| \leq |M| + |\#| + n - b - |M| - |\#| = n - b.$$

So x is compressible by b.

We have shown that any string x of length at least N that fails f is compressible by b. There are only finitely many strings of length less than N. Therefore only finitely many x that fail f can be incompressible by b. □