Information Theory **DCC/ICEx/UFMG**
Prof. Mário S. Alvim **2017/02**

SOLUTION OF HOMEWORK

Kolmogorov Complexity and Universal Probability
(Based on slide-set)

Necessary reading for this assignment:

- *Slide-set of Lecture 08 - Kolmogorov Complexity and Universal Probability*

Note: The exercises are labeled according to their level of difficulty: `[Easy]`, `[Medium]` or `[Hard]`. This labeling, however, is subjective: different people may disagree on the perceived level of difficulty of any given exercise. Don't be discouraged when facing a hard exercise, you may find a solution that is simpler than the one the instructor had in mind!

Review questions.

1. Answer formally the following questions:

   (a) Define the Kolmogorov Complexity of a string

   Instructor's solution: The Kolmogorov complexity of a string is the length of the shortest possible program that, when running in a Universal Turing Machine, outputs the string and then halts

   (b) When is a string considered truly random? Give an example of a truly random binary string and an example of a non-random string that looks random.

   Instructor's solution: A string is random, or algorithmically incompressible, if $K(s) \geq \ell(s)$ (i.e., if its Kolmogorov Complexity $K(s)$ is no smaller than its length).
   An algorithmically incompressible string has no regularities that can be exploited to make its description shorter.
   A binary string where each symbol is the result of the flip of a coin (0 for heads, 1 for tails) is truly random, because it is impossible to compress this string algorithmically.
   A non-random string that looks random is, for instance, the binary expansion of the decimal places of the number e.

   (c) What is the universal probability of a string? How is it related to the string's Kolmogorov complexity?

   Instructor's solution: The universal probability $p_{\mathcal{U}}(s)$ of a string s is the probability that s is produced as the output of an Universal Turing Machine fed with a random program: $p_{\mathcal{U}}(s) = \sum_{p:\mathcal{U}(p)=s} Pr(p) = \sum_{p:\mathcal{U}(p)=s} 2^{-\ell(p)}$.
   The universal probability of a string is related to its Kolmogorov complexity by the equation $p_{\mathcal{U}}(s) \approx 2^{-K(s)}$
   The intuition is that the shortest program that produces the string s will contribute exponentially more to the sum $\sum_{p:\mathcal{U}(p)=s} 2^{-\ell(p)}$ than all other programs that produce s (since the probability of programs decay exponentially with their length).

Problems.

2. (Cover & Thomas 14.1) `[Medium]` Let $x, y \in \{0,1\}^*$ be two binary sequences. Argue that the Kolmogorov complexity $K(xy)$ of the concatenation of x and y satisfies $K(xy) \leq K(x) + K(y) + c$.

   Instructor's solution: To describe the concatenation xy of the strings x and y it is enough to have a program that first describes the string x (which can be done with complexity at most $K(x)$ bits) and then describes the string y (which can be done with complexity at most $K(y)$ bits). Hence, the program for describing the concatenation xy will need at most $K(x) + K(y)$ bits to describe x and y, and a constant number of bits c to say that y must be printed right after x.

3. (Cover & Thomas 14.2) `[Medium]` Let n_1 and n_2 be two binary numbers.

   (a) Argue that the complexity $K(n_1 + n_2)$ of the sum of n_1 and n_2 satisfies $K(n_1 + n_2) \leq K(n_1) + K(n_2) + c$.

   Instructor's solution: To describe the sum $K(n_1 + n_2)$ we can use a program that generates n_1 (which can be done with complexity at most $K(n_1)$ bits), generates n_2 (which can be done with complexity at most $K(n_2)$ bits), and then has a instruction to add both numbers. Because the instruction to add has a constant size c bits, we have that $n_1 + n_2$ can be described in at most $K(n_1) + K(n_2) + c$ bits.

   (b) Give an example of binary numbers n_1 and n_2 that are complex, but such that $n_1 + n_2$ is simple.

   Instructor's solution: Consider the binary number n_1 consisting in N flips of a coin in which 0 represents heads and 1 represents tails, and the number n_2 representing the same N flips of a coin, but in which 0 represents tails and 1 represents heads.
   The sum $n_1 + n_2$ consists of a string of N 1s, which is very simple.

4. (Cover & Thomas 14.5 - *Monkeys on a computer*) `[Medium]` Suppose that a random program is typed into a computer. Give a rough estimate of the probability that the computer prints the following sequence:

   a) 0^n followed by any arbitrary sequence.

   Instructor's solution: A program $0n$ (in pseudo-language) that would write 0^n is the following.

```
program 0n {
    for i=1 to n do {
        print "0";
    }
}
```

   Note that the code of program `0n` above is a prefix of any code that writes 0^n followed by an arbitrary sequence, because by concatenating any further code this program we can only extend the output 0^n with an arbitrary sequence (including the empty sequence). Hence, by estimating program `0n`'s universal probability we are estimating the universal probability of any code that writes 0^n followed by an arbitrary sequence.
   Note that program `0n` uses 40 characters (spaces included, line returns ignored) to specify the program, and that the number n needs approximately $\log_{10} n$ decimal digits to be specified (e.g.,

to write 935 we need approximately $\log_{10} 935 = 2.97 \approx 3$ digts). Hence, for each n, the length of program `0n` is

$$\ell(\texttt{0n}) \approx (40 + \log_{10} n) \text{ characters} \approx (40 + 0.3 \log_2 n) \text{ characters.}$$

If we assume the program characters are encoded in ASCII, the code has about

$$\ell(\texttt{0n}) \approx (40 + 0.3 \log_2 n) \times 8 \text{ bits} \approx (320 + 2.4 \log_2 n) \text{ bits.}$$

bits in length, and hence we get an upper bound on its Kolmogorov complexity of

$$K(\texttt{0n}) \leq (320 + 2.4 \log_2 n) \text{ bits.}$$

Now, it follows that this program's universal probability is bounded by

$$\mathcal{U}(\texttt{0n}) \approx 2^{-K(\texttt{0n})} \geq 2^{-320-2.4 \log_2 n} \approx n^{-2.4} \cdot 2^{-320}.$$

Hence, the probability of a random program outputing 0^n followed by any arbitrary sequence can be estimated as roughly $n^{-2.4} \cdot 2^{-320}$.

b) $\pi_1 \pi_2 \ldots \pi_n$ followed by any arbitrary sequence, where π_i is the i-th bit in the expansion of π.

Instructor's solution: Algorithms for writing arbitrary digits of π are well-known. A quick search online shows that a C program `pi800` to write the first 800 digits of π can be written in 160 bytes = 1 280 bits (`https://crypto.stanford.edu/pbc/notes/pi/code.html`). Hence, we can estimate `pi800`'s Kolmogorov complexity as

$$K(\texttt{pi800}) \leq 1\,280 \text{ bits,}$$

and its universal probability as

$$\mathcal{U}(\texttt{pi800}) \approx 2^{-K(\texttt{pi800})} \geq 2^{-1\,280}.$$

Note, however, that if you want to find an estimate when the number of digits is $n \neq 800$, you'll need to refine the specification of your algorithm. We leave this as a little challenge for you!