Algorithmic Information Theory

**Algorithmic Information Theory** is a subfield of **Information Theory** and **Computer Science** that concerns itself with the relationship between **computation** and **information**.

**Discussion Topic 1.1**

- Data is simply a set of facts or figures, without any any organization/structure.

- By counting the number of facts, for instance (?)

- Information is data, but processed, organized and structured.

- Yes, for instance, entropy which is one of those measures, and measures the amount of uncertainty of a message.

- Computation is the act of processing data and turn it into information. This act is usually associated with (and performed by) computers.

- Yes, there are some problems that computers cannot solve. One obvious and well known example is the Halting Problem that concers with whether or not a program will ever terminate on a given input.

**Discussion Topic 1.2**

No, it is not. One bit of data doesn’t contains, at most, one bit of information. It may be the case, but it is not mandatory. For example, compressed data, definitely contains more then one bit of information per bit of data.

**Discussion Topic 1.3**

Yes, Bob is correct. This is because there 2M possible inputs of length M, but there are only 2M - 1 possible compressed outputs that are shorter than M. Thereforem, there must exist at least one compressed string that can’t be decompressed correctly.

**Discussion Topic 1.4**

- MI -> MII (Mx -> Mxx) -> MIII (Mx -> Mxx) -> MIIII (Mx -> Mxx) -> MUI (xIIIy -> xUy)

- No, I cannot, there is no rule that allows to either add letters in front of M (unless there is already one) and no rule that allows to exchange M of place as i tis, therefore, for any transformation applied to UIM, the letter M will always be the las tone, therefore i tis not possible to obtain the MIU sequence.

- For the same reason as before, no is not possible, M can never change place in the sequence if it is either at the beginning or at the end of the sequence.

- One possible set of steps are: MI -> MIU (xI -> xIU) -> MIUU (xI -> xIU) -> MIIUU (Mx -> Mxx) -> MIIIUU (Mx -> Mxx) -> MUUU (xIIIy -> xUy) -> MU (xUUy -> xy)

- No, there is no generic algorithm that answers “yes” or “no” to the question. If this was the case then the Halting Problem would have already been solved. However for some set of rules there are algorithms that can answer the question.

**Discussion Topic 1.5**

No, she isn’t, a computable function has a finite procedure, so there are only countably many computable functions. The set of finitary functions one the natural numbers is uncountable so most are not computable (for instance, the Halting Problem).

**Discussion Topic 1.6**

Hardly. While it is possible to toss a fair coin twenty times and in every single trial heads comes up, this is highly unlikely, it has a probability of 1 / 220 (= 1 / 1048576 ~= 9.54e-7). Yes, if the sequence had 10 ‘0’s and 10 ‘1’s it would be easier to trust. However, the coin could still not be fair.

**Discussion Topic 1.7**

Yes, but only if the biased coin doesn’t change its biased from one toss to another, this happens because even tho the probability of getting one of the sides is bigger than the other, the probability of getting heads first and then tails is the same as getting tails first and then heads, independently of the bias (as long as it doesn’t change).

**Discussion Topic 1.8**

Randomness is generally defined as the lack of pattern or predictability in events. It is usually associated with na uniform distribution, though a lot of times, we want to generate randomness based on a different distribution (for instance, gaussian). In classical probability theory measuring the randomness doesn’t make sense, since randomness refers to the process of obtaining the sequence, and not the sequence itself. If i tis refering to “algorithmic randomness” then yes, it is possible to determine.

**Discussion Topic 1.9**

He was simply trying to bring awareness about the fact that random generators are only pseudo random and not truly random, after all a strict arithmetical algorithm is to some point a deterministic one.

**Discussion Topic 1.10**

The phrase is a paradox, because it refers to that number, but the phrase only has 57 letters, therefore there must be an integer defined by this expression, but since the expression is self-contradictory (any integer it defines is definable in under sixty letters), there cannot be any integer defined by it.

**Discussion Topic 1.11**

I would say that, the program itself could be buggy, and therefore uncapable of correctly detecting whether another program would be bug-free or not.

**Discussion Topic 1.12**

This function is the implementation of the Collatz Conjecture, which to this day, has not be proven to exit for any number (although, most mathematicians believe, that, that is the case). However, since the function reveives a 32 bits integer, only a limited range of numbers can be passed to it, and for every number formed by 32 bits the function does return.

**Problem 2.1**

- At each second alice updates the flashlight state, and ensures, that it matches the coin tossing result (for example, on is heads, off is tails). On the other hand, Bob, registers the current flashlight state, at each second, once Bob has done this n times, he knows that a message has been deliver.

- Initially alice can make a combination of signals in order to tell Bob the length of the message, for instance, the first 16 seconds determine the length of the message, and Alice has set states at each second accordingly. After that they do the above procedure for n times, where n is the determined value in the first 16 seconds.

**Problem 2.2**

Using an identical procedure, but instead of 1 second be equivelant to a character, 5 seconds determine which character is being transmitted, therefore Alice has to update the flashlight once per second, in such a way, that after 5 seconds Bob can recognize the character that Alice is trying to transmit.

**Problem 2.3**

- Bob has to ask questions in such a way that he halves the set of possibilities, for instance, he may start by asking if the number is above 50, if not, than the possibilites become [1, 50], else they become [51, 100], he keeps repeting this process until he gets the answer. If b is the number of questions, then 2b = 100 (2 answer possible for each question). Hence, b = log2 100 ~= 6.644, meaning that on average Bob needs to do 6.644 to find the number.

- No, the minimum number of questions required is not the same. That number is calculated by the formula b = max\_random\_value – min\_ random\_value. Hence, b = 100 – 1 = 99.

**Problem 2.4**

- That’s due to the fact that in its binary format the maximum number doesn’t make use of all the available bits (basically, in order to give the theoretical number, the max value should be 2n where *n* is some positive value).

- Yes, it is (?)

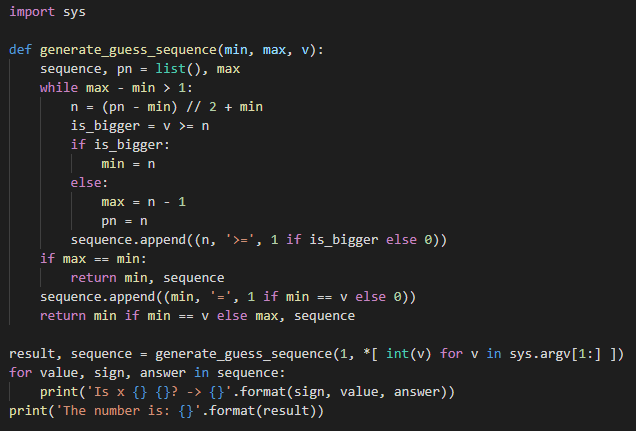
**Problem 2.5**

- The average amount of questions is 5/3.

- Just like before, the average amount of questions is 5/3.

- In case the numbers are not likely probable, then depending on the questions itself, the average changes accordingly (asking if the most probable is the chosen one in the first question, for instance, my give a lower average, then the other way around). Just like before, the theoretical value is different due to the fact that we are using a decimal base, and the amount of numbers is odd.

**Programming 2.1**



**Problem 2.6**

Using the idea above for each number we can set it in the right position comparing it using the previous method. Since the previous method has an average of log2 n (where n is the amount of numbers) amount of comparsions, and we do this comparsions for each number, meaning we do this n times, than on average it takes n log2 n which fits the lower bound case Ω (n log n).

**Problem 2.7**

On average, Bob needs to ask at least 0.5 + 1.5 log2100 questions, this is because, Bob will need to ask if she lied in the previous answer or not, and as soon as she answers yes, than Bob knows she can’t lie anymore, and so, he doesn’t need to ask anymore if she is lying or not, which means that in the worst case scenario he will ask 2 log2 100 questions, and in the best scenario, he will ask 1 + log2100 questions.

**Problem 2.8**

With n = 3, we can form 2 3 = 8 different 3-bit strings. Suppose that we pick the strings 000 and 111 as the possible codewords. Then, the set of 8 3-bit strings form a Hamming(3, 1) code (verify this). If each of the 3-bit strings occurs with equal probability, then there is a 1/4 probability of generating a correct codeword (2 out of 8) and a 3/4 probability of generating a erroneous codeword (6 out of 8). Therefore, the strategy for the game is the following:

1. If the player sees the other two players with the same hat color, then he assumes that he has a different hat color, because this would be the most probable case. Hence, he announces that color.
2. If the player sees the other two players with different hat colors, then, since in that case the probability of his hat having any of the two colors is the same, he passes.

Notice that, using this strategy, the probability of choosing the wrong color is still 50% each time someone announces a color. The gain is that the wrong guesses are not evenly distributed— from the total of twelve announcements associated to the eight combinations, the six that are wrong are concentrated in just two cases; all other six cases are error free.

**Homework 2.1**

Repeat above reasoning for Hamming(7, 4).

The problems that we have been addressing consider a measure of information that is usually called combinatorial, because it is only concerned with the number of possible objects involved, assuming that they occur with equal probability. Hence, for m distinct objects (a size-m alphabet), the amount of combinatorial information needed to specify each one is

Log2 *m* bits.

This approach for measuring information can be traced back at least to the works of Nyquist and Hartley:

* Harry Nyquist (1889–1976), Certain Factors Affecting Telegraph Speed, Bell System Technical Journal (Nyquist, 1924):

“This paper considers two fundamental factors entering into the maximum speed of transmission of intelligence by telegraph. These factors are signal shaping and choice of codes.”

* Ralph Hartley (1888–1970), Transmission of Information, Bell System Technical Journal (Hartley, 1928):

“A quantitative measure of “information” is developed which is based on physical as contrasted with psychological considerations.”

In his paper, Hartley proposes measuring the information content, H, of a message as

*H* = *n* log *m* = log *mn* ,

where m denotes the number of possible symbols (the size of the alphabet) and n the number of symbols in the message. So, Hartley defined the amount of information in a message as the logarithm of the number of possible messages (message space).

The base of the logarithm determines the unit of information used: hartley for base 10, nat for base e, bit for base 2.

**Problem 2.9**

52 log2 52 ~= 296.423 (?). Assign a bit sequence to each card and send it in order (?).