

COMPUTER SYSTEMS ARCHITECTURE - LECTURE 0x02

INTRODUCTION TO INFORMATION THEORY

Cristian Rusu, Bogdan Macovei

LAST TIME

- we saw how to convert from one numeral base to another
- two's-complement representation
- we saw addition and logical operations in binary
- but where do the bits on which we operate come from?
- today, we will study the concept of information itself ...

OUTLINE

- **information theory**
 - what information is
 - how we can measure the amount of information
 - data encoding
 - error detection and correction
- **bibliographic references**

INFORMATION THEORY

- **what is “information”?**
 - data that affects (usually reduces) the uncertainty we have about an event/phenomenon/etc.
- **an example: given a standard deck of 52 playing cards, which event brings the largest amount of information?**
 - we draw a card from the deck that is a heart
 - we draw a card from the deck that is a king
 - in which situation do we have more precise information about the card?

INFORMATION THEORY

- **what is “information”?**
 - data that affects (usually reduces) the uncertainty we have about an event/phenomenon/etc.
- **an example: given a standard deck of 52 playing cards, which event brings the largest amount of information?**
 - we draw a card from the deck that is a heart ($13/52$)
 - **we draw a card from the deck that is a king ($4/52$)**
- **why? the probability of drawing a king from the deck is much smaller, so if we draw this card we receive more information (rare events produce a lot of information)**
- **another way to see the problem: there are 13 hearts, but only 4 kings, so it is easier to guess which card was drawn**

INFORMATION THEORY

- how do we measure the amount of information?
- we introduce a random variable **X**:
 - this variable can take N distinct values ($x_1, x_2, \dots, x_N$)
 - each distinct value occurs with probability ($p_1, p_2, \dots, p_N$)
- how much information do we receive if we observe that the variable **X** took the value x_i ?

$$I(x_i) = \log_2 \left(\frac{1}{p_i} \right)$$

- the smaller p_i is, the larger the amount of information is
- in the previous example:

$$I(\text{king}) = \log_2 \left(\frac{1}{\frac{4}{52}} \right) = \log_2 \left(\frac{52}{4} \right) = 3.7 \text{ bits}$$

INFORMATION THEORY

- the previous cases:

$$I(\text{king}) = \log_2 \left(\frac{1}{\frac{4}{52}} \right) = \log_2 \left(\frac{52}{4} \right) = 3.7 \text{ biti}$$

$$I(\text{red card}) = \log_2 \left(\frac{1}{\frac{13}{52}} \right) = \log_2 \left(\frac{52}{13} \right) = 2 \text{ biti}$$

- how much information do we have if we know exactly which card was selected from the deck?

$$I(\text{a particular card}) = \log_2 \left(\frac{1}{\frac{1}{52}} \right) = \log_2 \left(\frac{52}{1} \right) = 5.7 \text{ biti}$$

- take a look at the values above: $5.7 = 3.7 + 2$
 - is this a coincidence?

INFORMATION THEORY

- some experiments do not provide complete information
- information can be accumulated: we can perform more experiments (we can ask more questions)
- information is neither created nor destroyed; it is constant
- we have used some concepts from probability theory
 - what is the probability that an event will occur?
 - $P(\text{event } A) = \frac{\text{number of outcomes in which } A \text{ occurs}}{\text{total number of outcomes}}$
 - $P(\text{event } A \text{ and event } B) = P(\text{event } A) \times P(\text{event } B)$
 - only if the two events are independent
 - the independence of events is assumed in many situations to simplify the results

INFORMATION THEORY

- suppose we have N equally likely events
- we run an experiment (imperfect from an informational point of view) that tells us that only M events were possible out of the total N
- e.g., $N = 52$ (because there are 52 possible cards), $M = 13$ (because we learn that we drew a heart, but we do not know exactly which card it is)
- how much information do we receive?

$$I = \log_2 \left(\frac{1}{M^{\frac{1}{N}}} \right) = \log_2 \left(\frac{N}{M} \right)$$

- what happens if:
 - $M = 1$
 - $N = M$
 - $M > N$

INFORMATION THEORY

- suppose we have N equally likely events
- we run an experiment (imperfect from an informational point of view) that tells us that only M events were possible out of the total N
- e.g., $N = 52$ (because there are 52 possible cards), $M = 13$ (because we learn that we drew a heart, but we do not know exactly which card it is)
- how much information do we receive?

$$I = \log_2 \left(\frac{1}{M \frac{1}{N}} \right) = \log_2 \left(\frac{N}{M} \right)$$

- **what happens if:**
 - $M = 1$ we reduced the uncertainty to a single option, perfect
 - $N = M$ we solved nothing; the uncertainty is the same
 - $M > N$ worse; the uncertainty is greater

INFORMATION THEORY

- **other examples**

- we flip a coin (heads/tails)
 - $N = ?$, $M = ?$, $I = ?$

$$I = \log_2 \left(\frac{1}{M^{\frac{1}{N}}} \right) = \log_2 \left(\frac{N}{M} \right)$$

INFORMATION THEORY

$$I = \log_2 \left(\frac{1}{M^{\frac{1}{N}}} \right) = \log_2 \left(\frac{N}{M} \right)$$

- **other examples**

- we flip a coin (heads/tails)
 - $N = 2, M = 1, I = \log_2 (2/1) = 1$ bit
- we draw a card and see that it is a heart
 - $N = ?, M = ?, I = ?$

INFORMATION THEORY

$$I = \log_2 \left(\frac{1}{M^{\frac{1}{N}}} \right) = \log_2 \left(\frac{N}{M} \right)$$

- **other examples**

- we flip a coin (heads/tails)
 - $N = 2, M = 1, I = \log_2 (2/1) = 1$ bit
- we draw a card and see that it is a heart
 - $N = 52, M = 13, I = \log_2 (52/13) = 2$ bits
- we roll two dice
 - $N = ?, M = ?, I = ?$

INFORMATION THEORY

$$I = \log_2 \left(\frac{1}{M^{\frac{1}{N}}} \right) = \log_2 \left(\frac{N}{M} \right)$$

- **other examples**

- we flip a coin (heads/tails)
 - $N = 2, M = 1, I = \log_2 (2/1) = 1 \text{ bit}$
- we draw a card and see that it is a heart
 - $N = 52, M = 13, I = \log_2 (52/13) = 2 \text{ bits}$
- we roll two dice
 - $N = 36, M = 1, I = \log_2 (36/1) = 5.17 \text{ bits}$
- in the “Batman” films, there is a villain who has a coin with “heads” on both sides
 - $N = ?, M = ?, I = ?$

INFORMATION THEORY

$$I = \log_2 \left(\frac{1}{M^{\frac{1}{N}}} \right) = \log_2 \left(\frac{N}{M} \right)$$

- **other examples**

- we flip a coin (heads/tails)
 - $N = 2, M = 1, I = \log_2 (2/1) = 1 \text{ bit}$
- we draw a card and see that it is a heart
 - $N = 52, M = 13, I = \log_2 (52/13) = 2 \text{ bits}$
- we roll two dice
 - $N = 36, M = 1, I = \log_2 (36/1) = 5.17 \text{ bits}$
- in the “Batman” films, there is a villain who has a coin with “heads” on both sides
 - $N = 1, M = 1, I = \log_2 (1/1) = 0 \text{ bits}$

ENTROPY

- **entropy**

- the average amount of information received about a variable X

$$H(X) = E(I(X)) = \sum_{i=1}^N p_i \log_2 \frac{1}{p_i} = \sum_{i=1}^N -p_i \log_2 p_i$$

- $H(X)$ is called the entropy of X
- $I(X)$ is the information about X
- E is “expected value”, the operation that computes the average value
- example: $X = \{A, B, C, D\}$ with probabilities $\{1/3, 1/2, 1/12, ?\}$

ENTROPY

- **entropy**

- the average amount of information received about a variable X

$$H(X) = E(I(X)) = \sum_{i=1}^N p_i \log_2 \frac{1}{p_i} = \sum_{i=1}^N -p_i \log_2 p_i$$

- $H(X)$ is called the entropy of X
- $I(X)$ is the information about X
- E is “expected value”, the operation that computes the average value
- example: $X = \{A, B, C, D\}$ with probabilities $\{1/3, 1/2, 1/12, 1/12\}$
 - what is the entropy?
 -

ENTROPY

- **entropy**

- the average amount of information received about a variable X

$$H(X) = E(I(X)) = \sum_{i=1}^N p_i \log_2 \frac{1}{p_i} = \sum_{i=1}^N -p_i \log_2 p_i$$

- $H(X)$ is called the entropy of X
- $I(X)$ is the information about X
- E is “expected value”, the operation that computes the average value

- example: $X = \{A, B, C, D\}$ with probabilities $\{1/3, 1/2, 1/12, 1/12\}$

$$H(X) = -\frac{1}{3} \log_2 \frac{1}{3} - \frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{12} \log_2 \frac{1}{12} - \frac{1}{12} \log_2 \frac{1}{12} = 1.626 \text{ bits}$$

- is the result “good”?

ENTROPY

- **entropy**

- the average amount of information received about a variable X

$$H(X) = E(I(X)) = \sum_{i=1}^N p_i \log_2 \frac{1}{p_i} = \sum_{i=1}^N -p_i \log_2 p_i$$

- $H(X)$ is called the entropy of X
- $I(X)$ is the information about X
- E is “expected value”, the operation that computes the average value

- example: $X = \{A, B, C, D\}$ with probabilities $\{1/3, 1/2, 1/12, 1/12\}$

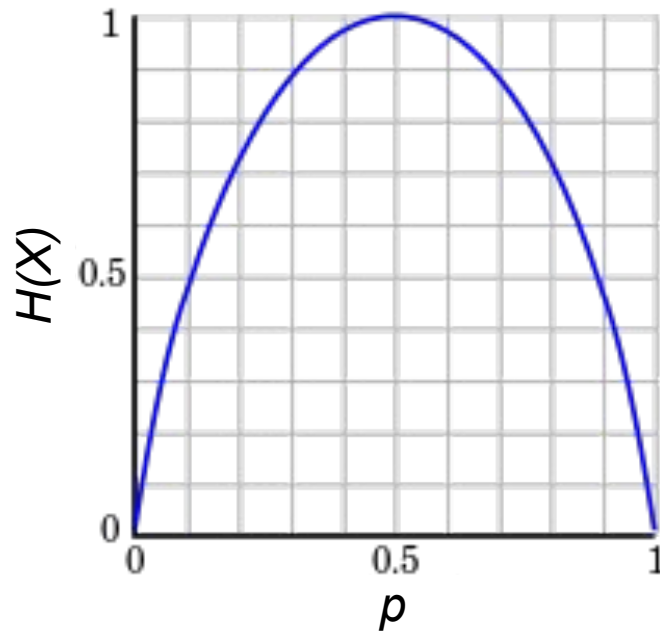
$$H(X) = -\frac{1}{3} \log_2 \frac{1}{3} - \frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{12} \log_2 \frac{1}{12} - \frac{1}{12} \log_2 \frac{1}{12} = 1.626 \text{ bits}$$

- the variable X has 4 options, so normally we would need 2 bits to store all possibilities, but (because the probabilities are not equal) we can encode using fewer than 2 bits

ENTROPY

$$H(X) = E(I(X)) = \sum_{i=1}^N p_i \log_2 \frac{1}{p_i} = \sum_{i=1}^N -p_i \log_2 p_i$$

- we consider a coin that we can flip
- the probability of seeing heads is p (and for tails it is $1-p$)



ENTROPY

- **example: $X = \{A, B, C, D\}$ with probabilities $\{1/3, 1/2, 1/12, 1/12\}$**
- **the entropy of X is 1.626**
- **what happens?**
 - I can store the variable X using 2 bits each (the encoding is: 00, 01, 10, 11) for each possible event
 - this is correct, but inefficient
 - why is it inefficient? because the entropy is 1.626, so it tells us that we can be more efficient; that is, the encoding above is not the most efficient way to encode the information
 - instead of 2 bits per event, we can use only 1.626 bits
 - entropy tells us that we cannot encode the variable above below 1.626 bits per event without losing information
 - for example, we could encode X with a single bit (0 or 1), but then we can distinguish only two events, not four
 - so we cannot “decode” what happened
- **entropy is the limit of possible compression**

DATA ENCODING

- **how do we reach that 1.626 bits for the four events?**
- **we use an encoding different from the standard one**
 - standard encoding $A = 00$, $B = 01$, $C = 10$, $D = 11$
 - with this encoding we have $ABBC = 00\ 01\ 01\ 10$
 - decoding is straightforward: we take 2 bits at a time and each pair is an event
- **another encoding (more efficient):**
 - variable code length: $A = 01$, $B = 1$, $C = 000$, $D = 001$
 - now we have $ABBC = 01\ 1\ 1\ 000$
 - now there are 7 bits, compared to 8 before (so it is better)
 - decoding must be unique! we must be able to go back
- **another encoding (even more efficient, but incorrect):**
 - variable code length: $A = 1$, $B = 0$, $C = 10$, $D = 11$
 - now we have $ABBC = 1\ 0\ 0\ 10$
 - 5 bits, even better
 - the problem? decoding: if we receive 10010, how do we decode it?

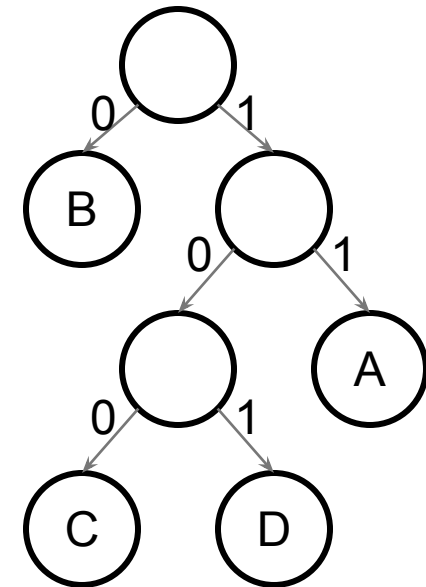
DATA ENCODING

- **how do we reach that 1.626 bits for the four events?**
- **we use an encoding different from the standard one**
 - standard encoding $A = 00$, $B = 01$, $C = 10$, $D = 11$
 - with this encoding we have $ABBC = 00\ 01\ 01\ 10$
 - decoding is straightforward: we take 2 bits at a time and each pair is an event
- **another encoding (more efficient):**
 - variable code length: $A = 01$, $B = 1$, $C = 000$, $D = 001$
 - now we have $ABBC = 01\ 1\ 1\ 000$
 - now there are 7 bits, compared to 8 before (so it is better)
 - decoding must be unique! we must be able to go back
- **another encoding (even more efficient, but incorrect):**
 - variable code length: $A = 1$, $B = 0$, $C = 10$, $D = 11$
 - now we have $ABBC = 1\ 0\ 0\ 10$
 - 5 bits, even better
 - the problem? decoding: if we receive 10010, how do we decode it?
 - $ABBC$ or CBC or ...

DATA ENCODING

- how can we create an efficient and unique encoding?

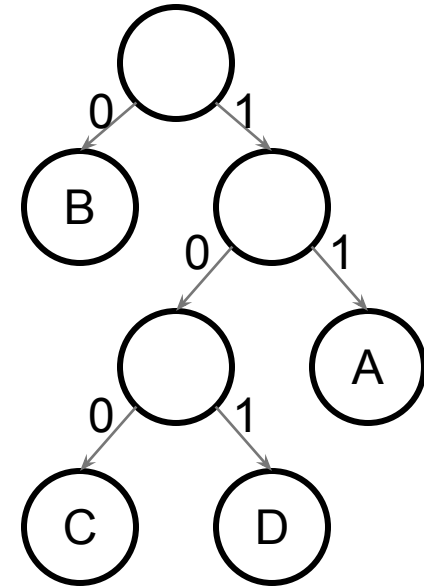
- a binary tree
 - the leaves are the codes
 - left/right is decided by 0/1
 - the encoding is:
 - B = 0
 - A = 11
 - C = 100
 - D = 101
 - this guarantees efficient encoding and unique decoding



- how do we generate the efficient encoding?
 - the Huffman algorithm
 - input: the probability of each event $\{1/3, 1/2, 1/12, 1/12\}$
 - output: the codes read from a binary tree (above)
 - the key: some events/symbols occur more often than others, so they receive a shorter encoding
 - if all events are equiprobable, then there is nothing we can do

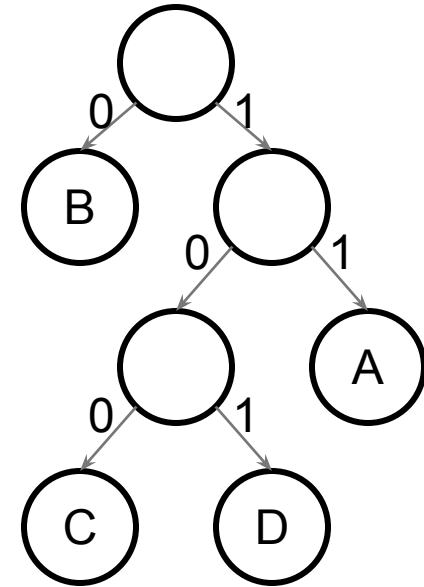
DATA ENCODING

- how can we create an efficient and unique encoding?
 - a binary tree
 - the leaves are the codes
 - left/right is decided by 0/1
 - the encoding is:
 - B = 0
 - A = 11
 - C = 100
 - D = 101
 - this guarantees efficient encoding and unique decoding
 - exercise: decode 00100111010011



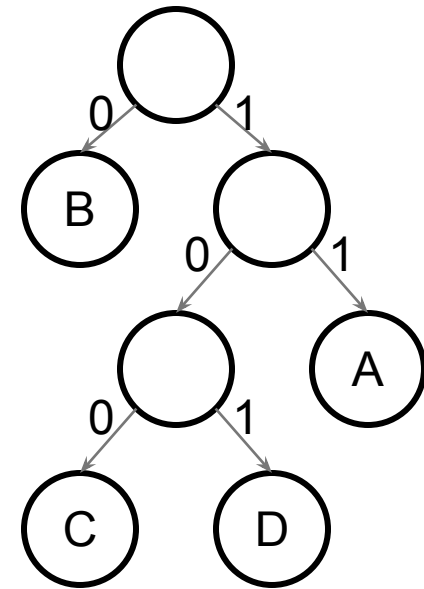
DATA ENCODING

- how can we create an efficient and unique encoding?
 - a binary tree
 - the leaves are the codes
 - left/right is decided by 0/1
 - the encoding is:
 - B = 0
 - A = 11
 - C = 100
 - D = 101
 - this guarantees efficient encoding and unique decoding
 - exercise: decode 0 0 100 11 101 0 0 11
 - solution: BBCADBBA



DATA ENCODING

- how can we create an efficient and unique encoding?
 - a binary tree
 - the leaves are the codes
 - left/right is decided by 0/1
 - the encoding is:
 - B = 0
 - A = 11
 - C = 100
 - D = 101
 - this guarantees efficient encoding and unique decoding
 - exercise: how do we compute the efficiency of this encoding? what is the average size of a message? the probabilities are $\{1/3, 1/2, 1/12, 1/12\}$

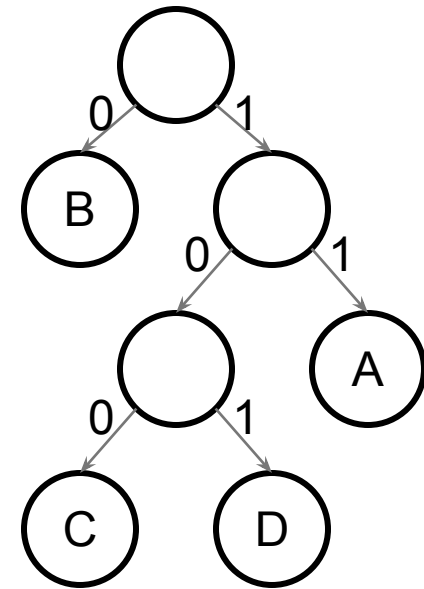


DATA ENCODING

- how can we create an efficient and unique encoding?

- a binary tree

- the leaves are the codes
- left/right is decided by 0/1
- the encoding is:
 - B = 0
 - A = 11
 - C = 100
 - D = 101
- this guarantees efficient encoding and unique decoding



- exercise: how do we compute the efficiency of this encoding? what is the average size of a message? the probabilities are $\{1/3, 1/2, 1/12, 1/12\}$
 - $2 \times 1/3 + 1 \times 1/2 + 3 \times 1/12 + 3 \times 1/12 = 1.667$ bits
 - compared to 1.626 bits, which is optimal

DATA ENCODING

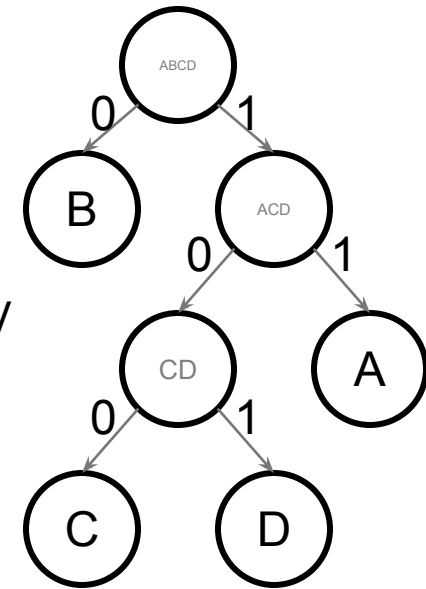
- **we have already seen numbers encoded in the binary system**
 - encoding/storing natural numbers and integers
 - what kind of encoding was this? fixed length
 - regardless of which number we store, we use N bits
 - what are the advantages of fixed-length encoding?
 - what are the advantages of variable-length encoding?

DATA ENCODING

- **we have already seen numbers encoded in the binary system**
 - encoding/storing natural numbers and integers
 - what kind of encoding was this? fixed length
 - regardless of which number we store, we use N bits
 - what are the advantages of fixed-length encoding?
 - it is simple; we know that each symbol has the same number of bits (so we know how much space we need, etc.)
 - we can directly access the i -th symbol in the string (ABBA etc.)
 - the implementation in electronic circuits is the same because all elements have the same number of bits (32 or 64 bits)
 - it is optimal if the symbols have equal probabilities
 - what are the advantages of variable-length encoding?
 - efficient encoding (storage space)

DATA ENCODING

- the **Huffman algorithm**
 - input: the probability of each event $\{1/3, 1/2, 1/12, 1/12\}$
- the **algorithm**:
 - take the least probable events
 - C and D, and put them in the tree
 - we create a new event CD; the probability of its occurrence is $1/6$ (the sum of C and D)
 - the new events are $\{A, B, CD\}$ with probabilities $\{1/3, 1/2, 1/6\}$
 - again, the least probable events
 - A and CD; A goes on the other leaf
 - the new events are $\{B, ACD\}$ with probabilities $\{1/2, 1/2\}$
 - again, the least probable events
 - now there are only two, $\{B, ACD\}$; B goes on the other leaf



DATA ENCODING

- the Huffman algorithm is optimal if we consider one symbol at a time
- but we can merge symbols, that is, build the same binary tree for all combinations of two symbols
 - AA, AB, AC, AD, BA, ..., CA, CB, CC, ..., DA, DB, DC, DD
- a few questions:
 - how many symbols do we have now? 16
 - what is the probability of the following events?
 - AA $\frac{1}{9}$
 - DD $\frac{1}{144}$
 - DB $\frac{1}{24}$
 - AD $\frac{1}{36}$
 - how do we compute the entropy now? a sum of 16 terms
 - the new average Huffman length, using pairs of symbols? 1.646 bits
 - better than before (Huffman with one symbol, 1.667 bits)
 - closer to the entropy of 1.626 bits (optimal)
 - therefore, if we group more symbols together, we converge to 1.626 bits

DATA ENCODING

- **we compress a text (we have a zip file, for example)**
 - we can compress: text, images, even executables
- **what stops us from compressing it again? (compressing the zip file)**

DATA ENCODING

- **we compress a text (we have a zip file, for example)**
 - we can compress: text, images, even executables
- **what stops us from compressing it again? (compressing the zip file)**
- **the contents of the zip file look completely random**
 - something perfectly compressed looks like noise
 - only noise cannot be compressed
 - if we lose something from the noise, we can no longer recover it
- **we encrypt a text**
- **hash for a text**

ERROR DETECTION / CORRECTION

- **error detection**

- what happens if we store a binary string but some error occurs: a 0 becomes 1 or a 1 becomes 0?

- for example:

- initially we have: 0110 0101
- the data is corrupted and we have: 1100 0001

- first step:

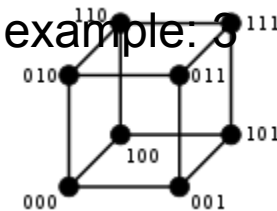
- we need to define a distance between the correct string and the corrupted one
- the Hamming distance between two binary strings: how many bits are different (bits in the same positions in the binary representation)
- the strings must have the same length
- the Hamming distance above: 3

ERROR DETECTION / CORRECTION

- error detection

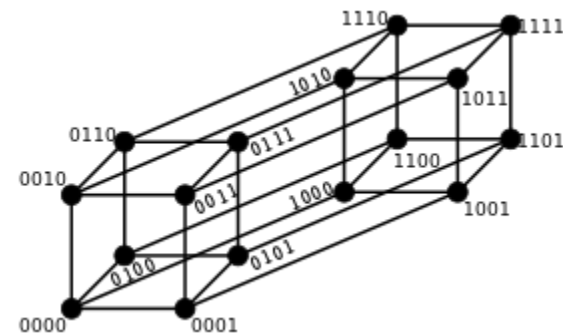
- first step:

- we need to define a distance between the correct string and the corrupted one
 - the Hamming distance between two binary strings: how many bits are different (bits in the same positions in the binary representation)
 - the strings must have the same length
 - the Hamming distance in the previous example: 3



- Hamming for 3-bit strings

- Hamming for 4-bit strings



ERROR DETECTION / CORRECTION

- **error detection**

- the previous example:

- initially we have: 0110 0101
 - the data is corrupted and we have: 1100 0001

- the problem: 0110 0101 and 1100 0001 are valid strings
 - that is, we cannot tell that an error occurred

- the idea: not all binary strings will be possible; in this way, if a binary string appears that is not allowed (that does not exist in our dictionary), then we know that an error occurred somewhere

- the simplest idea: add a parity bit to the symbols
 - 0 becomes 00
 - 1 becomes 11
 - only these two new symbols are valid

ERROR DETECTION / CORRECTION

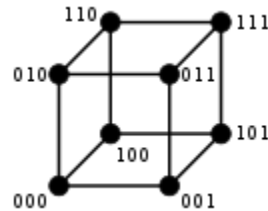
- **error detection**

- the simplest idea: add a parity bit to the symbols
 - 0 becomes 00
 - 1 becomes 11
 - only these two new symbols are valid
- if we receive 01 or 10, we know that an error occurred
- if we receive 00 or 11, we know that everything is OK
- what did we do? we added redundancy: that is, we can lose something and still figure out which symbol we have
- what do we lose? we are twice as inefficient (instead of one bit, we now need to store two bits)
- can we detect multiple errors? no. if there are two errors, then 00 can become 11 because of the errors

ERROR DETECTION / CORRECTION

- **error correction**

- a sufficiently large Hamming distance can also lead to error correction (not only error detection)
- let us consider 3-bit strings

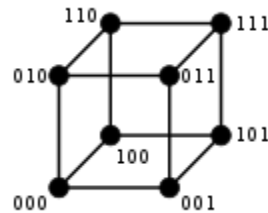


- the only valid codes are “000”, which is “0”, and “111”, which is “1”
 - if we receive “001” or “010” or “100”, we can suspect that it is “0”
 - if we receive “110” or “101” or “011”, we can suspect that it is “1”
- a Hamming distance of $2E+1$ can correct E errors
 - what is the intuition?

ERROR DETECTION / CORRECTION

- error correction

- a sufficiently large Hamming distance can also lead to error correction (not only error detection)
- let us consider 3-bit strings



- the only valid codes are “000”, which is “0”, and “111”, which is “1”
 - if we receive “001” or “010” or “100”, we can suspect that it is “0”
 - if we receive “110” or “101” or “011”, we can suspect that it is “1”
- a Hamming distance of $2E+1$ can correct E errors
 - what is the intuition? majority vote
- in general, if we want to detect E errors, we need a Hamming distance between codes of $E + 1$: that is, “0” can be “000” and “1” can be “111” – we are now 3 times less efficient in storage, but we can detect 2 errors and correct one error

INFORMATION THEORY

- an exercise ...

INFORMATION THEORY

Europe has recorded the largest weekly number of coronavirus cases so far, and the World Health Organization has warned that daily deaths could reach in April 2021 levels 4-5 times higher than in the spring of this year. Seventeen countries, including Russia, out of the 27 EU member states plus Great Britain are marked in red on the new European epidemic risk map.

INFORMATION THEORY

Europe has recorded the largest weekly number of coronavirus cases so far, and the World Health Organization has warned that daily death tolls could reach levels in April 2021 that are 4-5 times higher than in the spring of this year. Seventeen countries, including Romania, out of the 27 EU member states plus Great Britain, are marked in red on the new European epidemic risk map.

INFORMATION THEORY

- another exercise ...

INFORMATION THEORY

I cnduo't bveiee taht I culod aulacly uesdtannrd waht I was rdnaieg. Unisg the icndeblire pweor of the hmuan mnid, aocdcnig to rseecrah at Cmabrigde Uinervtisy, it dseno't mttar in waht oderr the lterets in a wrod are, the olny irpoamtnt tihng is taht the frsit and lsat ltter be in the rhgit pclae. The rset can be a taotl mses and you can sitll raed it whoutit a pboerlm. Tihs is bucseae the huamn mnid deos not raed ervey ltter by istlef, but the wrod as a wlohe. Aaznmig, huh? Yaeh and I awlyas tghhuot slelinpg was ipmorantt! See if yuor fdreins can raed tihs too.

INFORMATION THEORY

- why did we do these exercises?
- what is the “moral”?

INFORMATION THEORY

- why did we do these exercises?
- what is the “moral”?
- **Romanian and English are redundant**
 - in addition, the decoding machine we have (the brain) performs quite well in such situations
- we can extrapolate and say that all languages of the world are redundant
- otherwise, we could not communicate because of noise (in this case, noise is literal)
- **any communication/storage is redundant**

WHAT WE DID TODAY

- we discussed information
- we discussed Shannon entropy
- we saw how we encode/decode information
- error detection and correction

NEXT TIME ...

- digital abstraction
- we start discussing electronic circuits
- the transistor
- how operations on a computer system are implemented with electronic circuits
- basic circuits

FURTHER READING

- **David MacKay, Information Theory, Inference, and Learning Algorithms, 2003**
 - I Data Compression
 - the 3 introductory chapters
- **Huffman Codes: An Information Theory Perspective,** <https://www.youtube.com/watch?v=B3y0RsVCyrw>
- **The Universe is Hostile to Computers,** https://www.youtube.com/watch?v=AaZ_RSt0KP8
- **MacKay's videos (the first ones are the relevant ones for us)**
 - <https://www.youtube.com/playlist?list=PLruBu5BI5n4aFpG32iMbdWoRVA-A-Vcs06>
- **Ultimate Packer for Executables,** <https://en.wikipedia.org/wiki/UPX>

FURTHER READING(NOT INCLUDED IN THE EXAM)

- The Story of Information Theory: from Morse to Shannon to ENTROPY, <https://www.youtube.com/watch?v=rmBFaNgg4wk>
- Sean Carroll, The Biggest Ideas in the Universe | 20. Entropy and Information, <https://www.youtube.com/watch?v=rBPPOI5Ule0>
- Computerphile, playlist about entropy and information, https://www.youtube.com/playlist?list=PLzH6n4zXuckpKAj1_88VS-8Z6yr9zX_P6
- 3Blue1Brown, Hamming codes, how to overcome noise, <https://www.youtube.com/watch?v=X8jsijhlllA>
- 3Blue1Brown, Hamming codes part 2, the elegance of it all, https://www.youtube.com/watch?v=b3NxrZOu_CE
- Reed-Solomon Encoding
<https://www.youtube.com/watch?v=fBRMaEAFLE0>
<https://www.youtube.com/watch?v=xE4jEKx9fTM>

